Exploring Chaos

No. A-316
Instructional Computing Courseware
for Apple® II Series Computers

for the love of learning

mecc

Exploring Chaos

No. A-316
Instructional Computing Courseware
for Apple® II Series Computers

for the love of learning

This manual is compatible

with

the *Exploring Chaos* disk

Version 1.x

MECC warrants that it will, at its option, repair, replace, or return the purchase price of any product found by MECC to be defective in quality or for any other reason, and which is returned to MECC, postage paid. This warranty replaces and supersedes all other warranties stated or implied, including any warranty of design, merchantability, or fitness for a particular purpose or use. MECC will not be liable for any incidental, special, or consequential damages resulting from any use of or defects in the product.

MECC
6160 Summit Drive North
Minneapolis, Minnesota 55430-4003

ISBN 0-7929-0227-0

June 1992

Copyright © 1992 by MECC. All rights reserved. This software and manual may not be copied or transferred to another medium except as permitted by U.S. Copyright Law or as authorized by MECC. Apple® is a registered trademark of Apple Computer, Inc.

Library of Congress Cataloging-in-Publication Data

Exploring chaos [computer file]. — Version 1.0.

1 computer disk : sd., col.; 3.5 in. + 1 manual.
System requirements: Apple II series; 128K RAM; ProDOS; BASIC; 1 floppy disk drive; monochrome or color monitor (color highly recommended); Apple-compatible printer (optional).
Title from title screen.
Ed. statement from disk label.
Copy-protected.
Audience: Grades 7–12.
Issued also on 5.25 in. computer disk.
Summary: Introduces students to new topics in the fields of dynamic systems, fractal geometry, and chaos theory. Using a game-playing environment, students learn about cellular automata, random walks, iteration, and random fractals.
"A-316"—Disk label.
1. Iterative methods (Mathematics)—Juvenile software. 2. Chaotic behavior in systems—Juvenile software. 3. Fractals—Juvenile software. [1. Chaotic behavior in systems—Software. 2. Fractals—Software. 3. Iterative methods (Mathematics)—Software.] I. Minnesota Educational Computing Corporation.
Q172.5.C45 511.3 92-20617
ISBN 0-7929-0227-0

MECC Software License Information for Apple II Products

The information below pertains only to the software and documentation accompanying this notice. The rights and obligations you have with regard to other MECC products may differ from those described below.

> Software piracy is a crime. According to U.S. Copyright Law, illegal reproduction of software can be subject to civil damages of up to $100,000 and criminal penalties, including fines and imprisonment. You should carefully read the terms and conditions of this agreement before using this package. Use of any portion of this package indicates your agreement to these terms and conditions. If you do not agree to these terms, return the package to your place of purchase.

The rights and obligations you have regarding the use of this software package depend on whether or not you are a current MECC Apple II Member. MECC Apple II Members have license agreements with MECC that grant them rights that non-members do not have. If you are not sure if you are currently a MECC Apple II Member, contact your computer coordinator or MECC to determine your status.

CURRENT MECC APPLE II MEMBERS ONLY

If you are currently a MECC Apple II Member, this section applies to you.

Permitted Uses

You may:

- make unlimited copies of this software and its documentation, provided that:
  (1) all copies of disks are made using a current and legal MECC Copy System,
  (2) an official MECC label containing information provided by the MECC Label Utility is affixed to each disk that is copied,
  (3) only complete copies of manuals are distributed, with the exception of student worksheets;
- load and operate the software on more than one computer at a time, provided that the software allows such use;
- install this software on a file server and run the software on a file server, provided that you have purchased the appropriate MECC file-server installation software (network utility).

Prohibited Uses

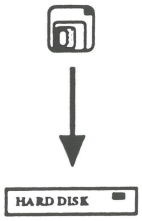
You may not:

- use this software in any locations except those specified by your Member License Agreement;
- transfer the software and all rights under this agreement to another person or entity;
- modify the software except as permitted in the normal use of the product;
- reverse engineer, disassemble, decompile, or make any attempt to discover the source code of the software;
- translate or create derivative works based on the software;
- remove, obscure, or alter any notice of the MECC software copyright or other proprietary rights related to the package;
- sub-lease, sell, lend, rent, or lease any portion of the software;
- copy any portion of the software except as permitted by your Member License Agreement;
- transfer the software or any direct product to any person or entity in violation of the United States Export Administration Act.

Copy Privileges

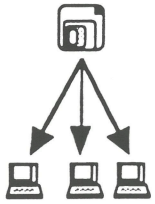
Provided that you use a current, legal MECC Copy System, you may make as many copies of the MECC software as you like.

Local Hard-Disk Privileges



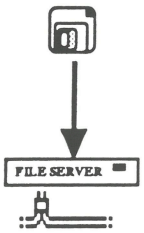
You may load the software on a local hard disk if you purchase the appropriate MECC utility.

Multiple-Loading Privileges



If the software permits such use, you may load the software onto more than one computer at a time ("load and go").

File-Server Privileges



You may load the software on a file server if you purchase the appropriate MECC file server installer software.

NON-MECC APPLE II MEMBERS

If you are *not* a MECC Apple II Member, this section applies to you.

Permitted Uses

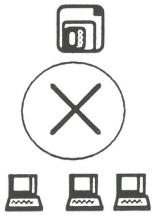
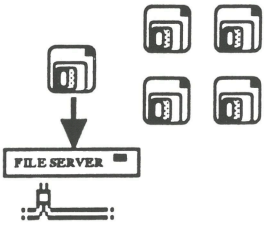
You may:

- make a backup copy of the software for archival purposes;
- install this software on a file server and execute the software on a network on more than one computer, provided that the total number of computers simultaneously using the software does not exceed the number of copies you have purchased, and that you use the appropriate MECC file-server installation software;
- transfer the software and all rights under this agreement to another person along with a copy of this agreement, provided that you supply MECC with written notice and the other party reads and accepts this agreement.

Prohibited Uses

You may not:

- load and operate the software on more than one computer at a time;
- modify the software except as permitted in the normal use of the product;
- reverse engineer, disassemble, decompile, or make any attempt to discover the source code of the software;
- translate or create derivative works based on the software;
- remove, obscure, or alter any notice of the MECC software copyright or other proprietary rights related to the package;
- sub-license, sell, lend, rent, or lease any portion of the software;
- copy any portion of the software except as permitted for archival purposes or transfer to another medium;
- transfer the software or any direct product to any person or entity in violation of the United States Export Administration Act.

Copy Privileges	Local Hard-Disk Privileges	Multiple-Loading Privileges	File-Server Privileges
			
You may make a single copy to be used for archival purposes only.	You may load the software on a local hard drive if you purchase the appropriate MECC utility.	You may not load the software onto more than one computer at a time.	You may install this software on a file server and use it on more than one computer if the total number of computers simultaneously using the software does not exceed the number of copies that you have purchased.

Limited Warranty

ProDOS products: MECC warrants that the accompanying disks will operate on an Apple IIe or later computer with 128K of RAM. GSOS Products: MECC warrants that the accompanying disks will operate on an Apple IIGS or later computer with 1M of RAM and appropriate system software. You may return this package with proof-of-purchase within 30 days for a full refund—no questions asked. For 90 days after purchase, MECC will replace free of charge any defective original disk. After 90 days, MECC will replace a defective disk at a nominal charge for the lifetime of the product (three years following the product's discontinuation). ***Except as specifically set forth above, MECC disclaims all other warranties, either expressed or implied, including but not limited to implied warranties of merchantability or fitness for a particular purpose. This limited warranty gives you specific legal rights, and you may also have other rights, which vary from state to state.*** Except as specifically set fourth above, MECC and its distributors will in no event be liable for any damages whatsoever, either direct or indirect, including but not limited to loss of business profits, income, or use of data. Some states do not allow exclusion or limitation of incidental or consequential damages; therefore, the limitations set forth in this agreement may not apply to you. ***Apple Computer, Inc. ("Apple") makes no warranties, expressed or implied, including without limitation the implied warranties of merchantability and fitness for a particular purpose, regarding the Apple software. Apple does not warrant, guarantee, or make any representations regarding the use of or the result of the use of the Apple software in terms of its correctness, accuracy, reliability, currentness, or otherwise. The entire risk as to the results and performance of the Apple software is assumed by you. The exclusion of implied warranties is not permitted by some states. The above exclusion may not apply to you. In no event will Apple, its directors, officers, employees or agents be liable to you for any consequential, incidental, or indirect damages (including damages for loss of business profits, business interruption, loss of business information, and the like) arising from the use of or the inability to use the Apple software even if Apple has been advised of the possibility of such damages. Because some states do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitations may not apply to you.*** Apple's liability to you for actual damages from any cause whatsoever, and regardless of the form of the action (whether in contract, tort [including negligence], product liability, or otherwise), will be limited to $50.

Government End Users

If this software is acquired by or on behalf of a unit of the United States Government, this provision applies: The software (a) was developed at private expense, is existing computer software, and was not developed with government funds; (b) is a trade secret of MECC for all purposes of the Freedom of Information Act; (c) is "restricted computer software" submitted with restricted rights in accordance with subparagraphs (a) through (d) of the Commercial Computer Software-Restricted Rights clause at 52.227-19 and its successors; (d) in all respects is proprietary data belonging solely to MECC; (e) is unpublished and all rights are reserved under copyright laws of the United States. For units of the Department of Defense ("DoD"), this software is licensed with "Restricted Rights" as that term is defined in the DoD Supplement to the Federal Acquisition Regulation 52.227-7013 (c) (1) (ii), Rights and Technical Data and Computer Software and its successors, and: Use, duplication, or disclosure is subject to restrictions as set forth in subdivision (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at 52.227-7013. MECC, 6160 Summit Drive North, Minneapolis, Minnesota 55430-4003. If this software was acquired under GSA schedule, the U.S. Government has agreed to refrain from changing or removing any insignia or lettering from the Software or the documentation that is provided or from producing copies of the manuals or media (except for backup purposes and in accordance with the terms of this Agreement).

Table of Contents

The Product at a Glance 1
- **Mosaic Madness** 2
- **The Life Game** 3
- **Random Walk** 4
- **Sticky Dots** 5
- **The Chaos Game** 6

The Product in Detail 7
- **Mosaic Madness** 7
- **The Life Game** 10
- **Random Walk** 16
- **Sticky Dots** 21
- **The Chaos Game** 24
- **Randomness Test** 29

Management Options 31

Classroom Ideas 35
- **Mosaic Madness** 35
- **The Life Game** 37
- **Random Walk** 40
- **Sticky Dots** 42
- **The Chaos Game** 43

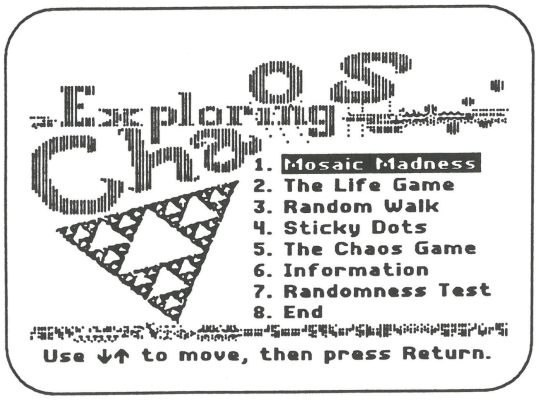
Student Worksheets 45

Suggested Readings 49

Thinking Skills 50

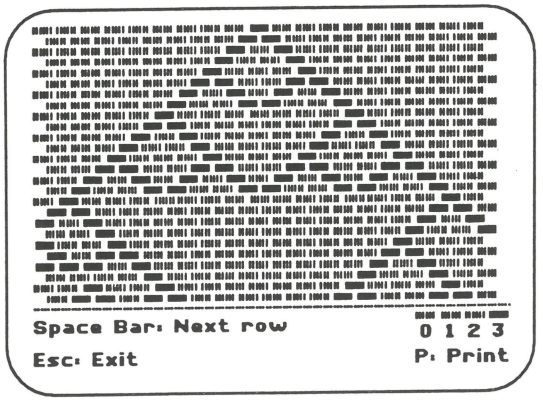
Credits 53

The Product at a Glance



Exploring Chaos introduces students to the contemporary topics of chaos and fractals, which are part of the study of dynamic systems. Activities include investigating cellular automata, playing the classic game of Life, designing random walks, and generating fractal images. The five programs in this package are designed in an open-ended manner to encourage exploration and discovery learning. An information section and a test to validate the randomness of the random-number generator used in each program are also included.

Title:	*Exploring Chaos*
Subject:	Mathematics
Topics:	Fractals; chaos; dynamic systems
Grade Range:	7–12
Program Type:	Simulation; discovery learning
Required Hardware:	Apple II series computer with 128K; color monitor recommended (required for **Mosaic Madness** program); printer is optional
Classroom Use:	Individual students, small group, demonstration
Learning Objectives:	• to explore a linear cellular automaton • to explore a two-dimensional cellular automaton • to explore the random-walk concept • to explore random fractals • to explore deterministic fractals



Mosaic Madness

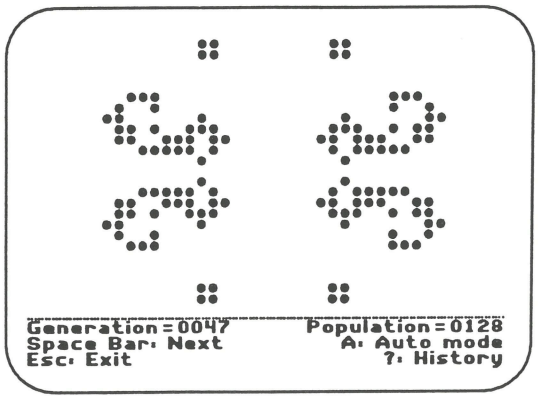
In **Mosaic Madness,** students generate a variety of patterns using its simple one-dimensional cellular automaton and discover the role of initial conditions and rules in producing various types of patterns. A color monitor is required.

Topic: Cellular automata

Program Type: Simulation; discovery learning

Learning Objectives:

- to explore the concept of linear cellular automata
- to determine the roles played by initial conditions (top row of tiles) and the rules for coloring
- to discover the effect of minor changes in initial conditions
- to discover the different types of patterns that can be produced



The Life Game

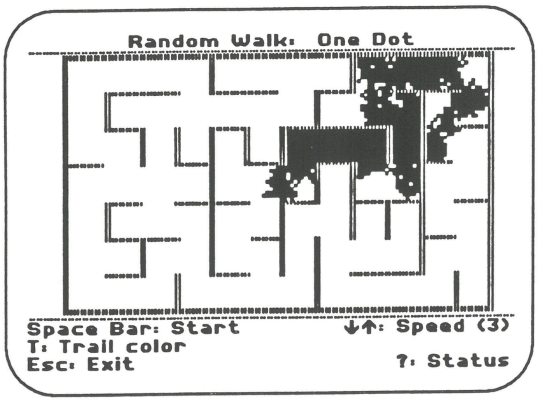
In **The Life Game**, students observe the behavior of various colonies of cells in a two-dimensional cellular automaton. Students use the original, classic rules or make up their own as they try to discover some new and unique "forms of life."

Topic: Cellular automata (two-dimensional)

Program Type: Simulation; discovery learning

Learning Objectives:

- to explore the concept of a two-dimensional cellular automaton
- to become familiar with the original game of Life and to identify various types of life forms (gliders, blinkers, etc.)
- to discover the effect of minor changes in initial conditions
- to explore the role of the rules



Random Walk

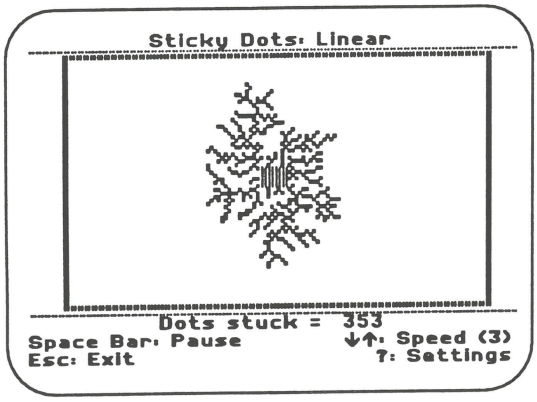
In **Random Walk**, students witness the behavior of a dot randomly wandering about inside a box and try to predict if it will hit a wall, the number of steps it will take before reaching a certain spot, or if it will make it through a maze.

Topic: Random walk

Program Type: Simulation; discovery learning

Learning Objectives:

- to introduce students to the concept of a random walk
- to allow students to experiment with the variables that influence the outcome of a random walk



Sticky Dots

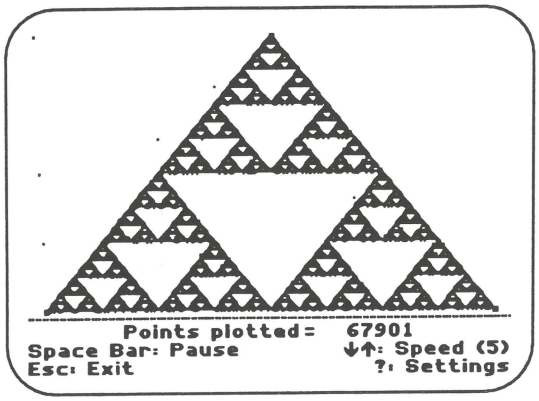
In **Sticky Dots,** based on the idea of a random walk, students explore one method of producing a random fractal. The process starts with a cluster of "sticky stuff" known as an aggregate. Random dots that wander too close to the aggregate stick to it and become part of it. Students choose between using either random dot movement or linear ballistic movement.

Topics: Diffusion-limited aggregation; random fractals

Program Type: Simulation; discovery learning

Learning Objectives:

- to introduce students to diffusion-limited aggregation
- to introduce students to ballistic aggregation
- to compare fractals produced under different conditions



The Chaos Game

In **The Chaos Game**, students explore an interesting environment that uses randomness to generate an image known as a deterministic fractal. Students can produce a variety of images by adjusting any of the variables controlling the process.

Topics: Randomness; iteration; deterministic fractals; self-similarity

Program Type: Simulation; discovery learning

Learning Objectives:

- to become familiar with the Chaos game
- to generate various deterministic fractals
- to change variables in the process and compare results

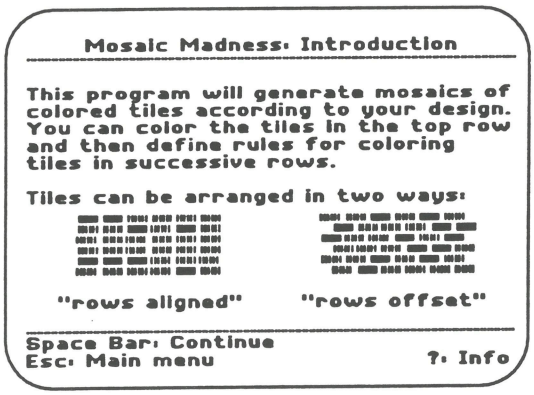
The Product in Detail

Mosaic Madness

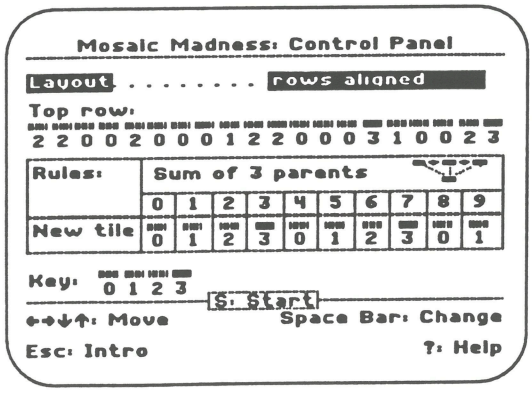
Mosaic Madness is an example of a cellular automaton, that is, a system in which the appearance or behavior of an individual cell is dictated, according to some set rules, by cells in the prior generation. This concept was first introduced in the 1950s by John von Neumann and Stanislaw Ulam, who wanted to design a machine that could self-replicate itself again and again. Today, with the help of modern computers, cellular automata are used in an attempt to model a variety of physical processes, such as the flow of fluids, the spread of diseases, and the path of raging forest fires.

This program represents a linear or one-dimensional cellular automaton, a rather new type of cellular automaton developed by Stephen Wolfram in 1984. The simulation begins with an initial row containing several colored cells. Each color has an assigned value. A new row of cells is then generated according to a set of coloring rules. To determine the color of each new cell in the next generation, the values of selected parent cells are added together. A table is provided that indicates what color is produced by that particular sum. In order to view the results, each new row is displayed directly beneath the previous row. In this way, one can view the pattern, or lack of a pattern, that is produced.

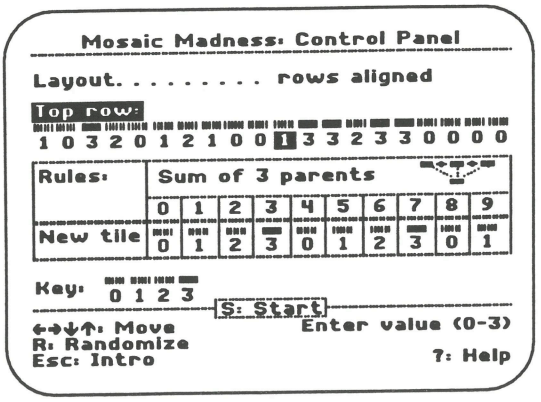
Models of this type can become quite complex. For example, the initial row can contain hundreds of cells of many colors, and the rules can depend on finding the sum of several cells on each side of any given cell. **Mosaic Madness** will generally remain quite simple; however, because color is used, a color monitor is needed to effectively use the program.


Mosaic Madness: Introduction
This program will generate mosaics of colored tiles according to your design. You can color the tiles in the top row and then define rules for coloring tiles in successive rows.
Tiles can be arranged in two ways:
"rows aligned"
"rows offset"
Space Bar: Continue
Esc: Main menu
?: Info

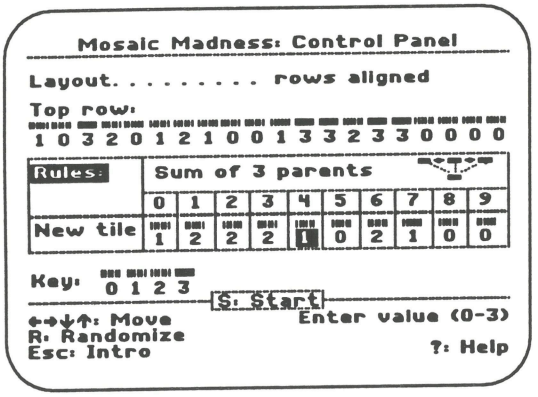
The opening screen introduces the program and shows examples of the two ways that the rows can be arranged. Additional background information is also available at this point by pressing the ? key.


Mosaic Madness: Control Panel
Layout . . . . . rows aligned
Top row:
2 2 0 0 2 0 0 0 1 2 2 0 0 0 3 1 0 0 2 3
Rules:
Sum of 3 parents
0 1 2 3 4 5 6 7 8 9
New tile
0 1 2 3 0 1 2 3 0 1
Key:
0 1 2 3
S: Start
←→↓↑: Move
Space Bar: Change
Esc: Intro
?: Help

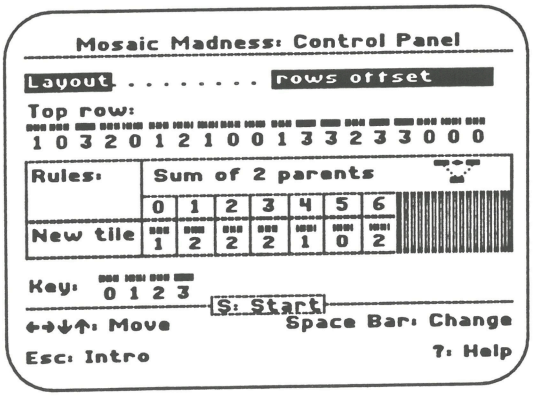
The next screen is called the Control Panel. Here you can change any of the three program variables: row layout, top row colors, and the rules for coloring new tiles. Choosing the "rows aligned" format will produce a checkerboard-style mosaic, while the "rows offset" format will produce a mosaic with a brick-wall design (see screen above).


Mosaic Madness: Control Panel
Layout. . . . . rows aligned
Top row:
1 0 3 2 0 1 2 1 0 0 1 3 3 2 3 3 0 0 0 0
Rules: Sum of 3 parents
0 1 2 3 4 5 6 7 8 9
New tile 0 1 2 3 0 1 2 3 0 1
Key: 0 1 2 3
S: Start
Enter value (0-3)
←→↓↑: Move
R: Randomize
Esc: Intro
?: Help

Moving the cursor down to the "Top row" section allows you to change the colors of the tiles in the top row. You can change the color of any tile by moving the highlight cursor to the tile and pressing the appropriate number (0–3) for the desired color. Another feature is to have the colors in the top row "randomize" by simply pressing the R key.

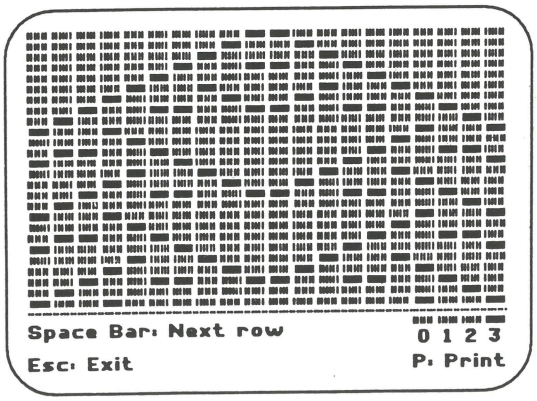

Mosaic Madness: Control Panel
Layout. . . . . rows aligned
Top row:
1 0 3 2 0 1 2 1 0 0 1 3 3 2 3 3 0 0 0 0
Rules: Sum of 3 parents
0 1 2 3 4 5 6 7 8 9
New tile 1 2 2 2 1 0 2 1 0 0
Key: 0 1 2 3
S: Start
Enter value (0-3)
←→↓↑: Move
R: Randomize
Esc: Intro
?: Help

The "Rules" section allows you to modify the rules for determining new tile colors. The "rows aligned" format uses the sum of three parent tiles in the row above to determine the color of each new tile. Since the largest color value of any tile is 3, the maximum sum for three tiles is 9. Therefore, the table contains color assignments for all sums in the range of 0 to 9. These values can be entered directly (0–3 keys) or randomly assigned by pressing the R key.

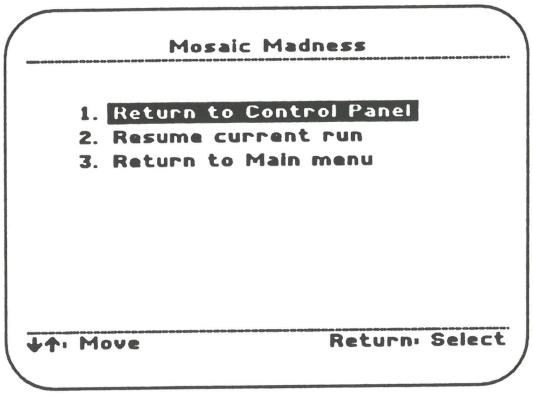

Mosaic Madness: Control Panel
Layout. . . . . rows offset
Top row:
1 0 3 2 0 1 2 1 0 0 1 3 3 2 3 3 0 0 0
Rules: Sum of 2 parents
0 1 2 3 4 5 6
New tile 1 2 2 2 1 0 2
Key: 0 1 2 3
S: Start
←→↓↑: Move
Space Bar: Change
Esc: Intro
?: Help

When "rows offset" is selected, the sum of only two parent tiles is used to determine the color of each new tile. The maximum sum is now reduced to 6, and the table is reduced accordingly.

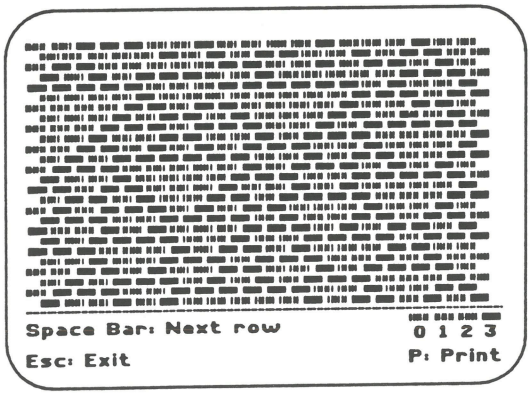
Note: With offset rows, tiles on the left or right edges have only one parent tile in the previous row. With aligned rows, each edge tile has only two parents in the previous row.



When you are satisfied with the initial conditions (top row) and the rules, press the S key to start generating the mosaic. The top row will appear at the very top of the screen. Each new row is then displayed by pressing the Space Bar. At any time while displaying the mosaic, you can print the screen, along with a copy of the current Control Panel. If you have a color printer, patterns can be seen. With a regular printer, some patterns will be difficult to see.



After you are done viewing the mosaic, press the Escape key and this menu will appear. You can return to the Control Panel and make changes, resume adding to the mosaic, or return to the Main menu.



If you return to the Control Panel and make changes to the top row or to the rules, new patterns (or, in some cases, no patterns) will emerge. Can you discover the roles played by the top row and the rules? Can you discover the different classes of patterns? Can you produce some specified pattern? Explore these questions and others related to linear cellular automata.

Some suggestions for using this activity in a classroom setting are provided in the "Classroom Ideas" section (pages 35–36).

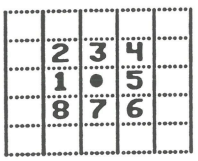
The Life Game

The game of Life was invented by John Conway, a mathematician at the University of Cambridge in Great Britain. His goal was to develop a model with a very simple set of rules that would result in a population with unpredictable behavior. The original game was played on a checkerboard where black markers were arranged to represent a starting colony of living organisms. A set of "genetic laws" was then used to determine which markers would survive, which would die, and where new births would take place. These rules were based on the number of neighbors surrounding a cell, which could total as many as 8. To determine the next generation of life, the process went something like this:

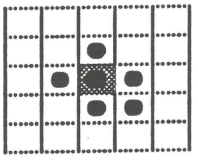
1. All markers with exactly 0 or 1 neighbor die due to isolation. All markers with 4 or more neighbors die due to over-population. Locate these markers and place another black marker on top of them to denote their location.
2. Locate all empty cells that have exactly 3 neighbors and place a white marker in the cell. These locations denote where births will take place.
3. Double check to make sure you have made no mistakes. Then remove all piles of two markers (representing deaths) and replace all white markers (representing births) with black markers. You have now produced the next generation of life.
4. Repeat this process and observe the behavior of the markers over a period of several generations.

It soon became obvious that a computer could speed up and greatly enhance this process! When the game was introduced to the public in October 1970 by Martin Gardner in the "Mathematical Games" column in *Scientific American*, it quickly caught the interest of many computer scientists and mathematicians. In the article, Conway offered a $50 prize to anyone that could disprove his conjecture that no pattern in the game could grow without limit. For example, Conway suggested trying to design a "gun" that could shoot out moving objects or a "puffer train" that would leave behind a trail of smoke. In his column a few month later, Gardner announced that the prize had been won by a group in M.I.T.'s Artificial Intelligence Project for discovering the "glider gun." The group made many other interesting discoveries, including the placement of "glider guns" so that the intersecting streams of "gliders" built a factory, which then assembled and fired a "lightweight spaceship" every 300 generations! The game of Life quickly established itself as a favorite pastime and an intriguing programming assignment.

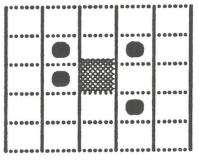
An understanding of the gameboard and the neighborhoods is very important in learning how the game is played. Conway's game calls for an infinite grid of square cells. However, most new players have to settle on using a checkerboard or a large sheet of graph paper. Each square cell has exactly eight neighboring cells (see diagram below). To denote that a cell contains a living organism, a marker or dot is placed in the appropriate cell. When we refer to the "neighbors" surrounding any cell, we mean the number of neighboring cells that contain life.



Every cell has 8 neighboring cells.



A living cell with 5 neighbors.

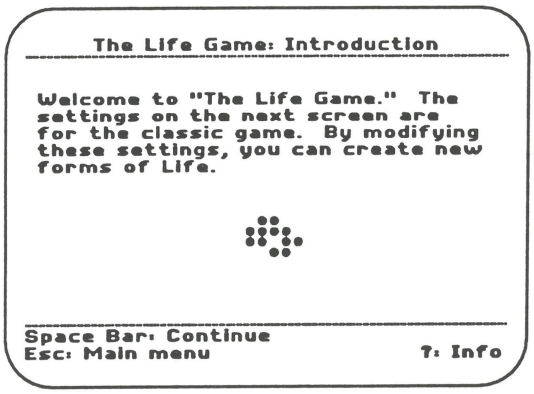


An empty cell with 4 neighbors.

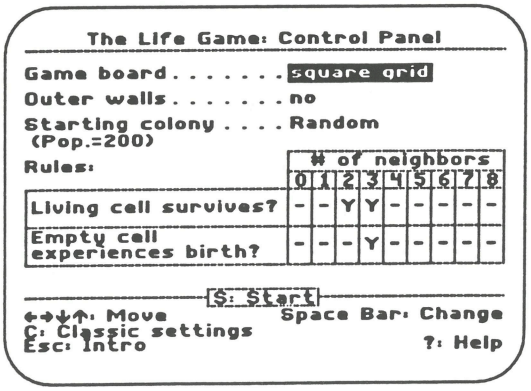
You play **The Life Game** by watching the evolution of a starting colony of dots over several generations. The rules for finding the next generation are:

1. All living cells with exactly 0 or 1 neighbor die (isolation).
2. All living cells with exactly 2 or 3 neighbors survive (ideal conditions).
3. All living cells with 4, 5, 6, 7, or 8 neighbors die (over-population).
4. All empty cells with exactly 3 neighbors experience a birth.
5. Births and deaths occur simultaneously.

It is worth noting that Conway experimented with many different gameboards and hundreds of different rule combinations before choosing the settings just discussed. Nearly all Life programs use only these "classic" settings. This program, however, allows you to modify the gameboard and even change the rules in an attempt to find new patterns and unusual behaviors. A simple editor is provided to allow you to quickly create or edit starting colonies. Now you can not only play the classic game, but you can also make up your own set of rules for Life!

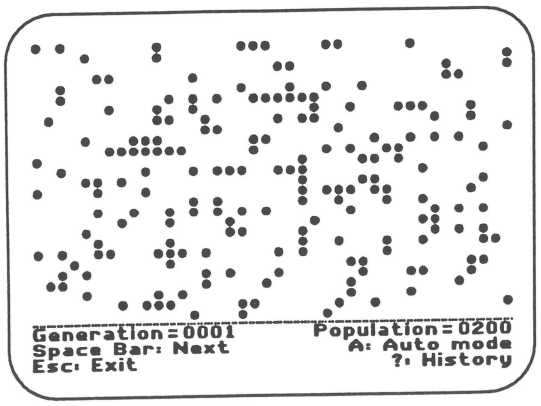

The Life Game: Introduction
Welcome to "The Life Game." The settings on the next screen are for the classic game. By modifying these settings, you can create new forms of Life.
Space Bar: Continue
Esc: Main menu
?: Info

The opening screen introduces the program and shows an interesting pattern known as a "lightweight spaceship," first discovered by Conway. Background information about the game is available at this point by pressing the ? key.

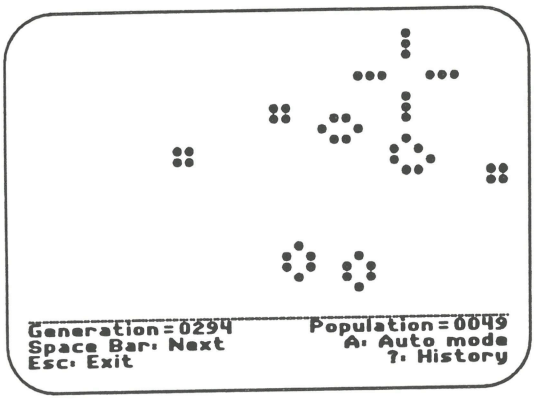

The Life Game: Control Panel
Game board . . . . . square grid
Outer walls . . . . . no
Starting colony . . . . Random
(Pop.=200)
Rules:
# of neighbors
0 1 2 3 4 5 6 7 8
Living cell survives? - - Y Y - - - - -
Empty cell experiences birth? - - - Y - - - - -
S: Start
←→↓↑: Move
Space Bar: Change
C: Classic settings
Esc: Intro
?: Help

The next screen is the Control Panel. When first entering the program, settings are for Conway's classic game and a starting colony of 200 randomly placed dots. You can change any of the game settings or just start with the current settings by pressing the S key.

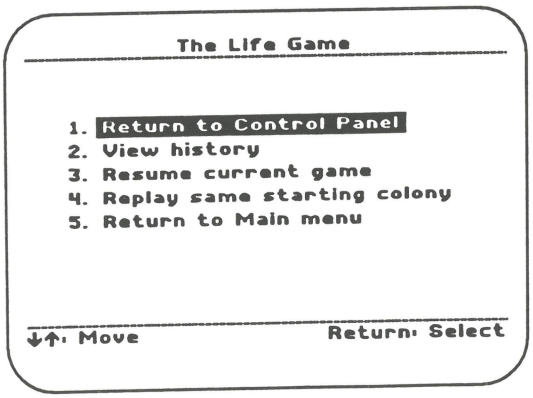
Let's run through a game using these default settings. Press the S key to start the game.



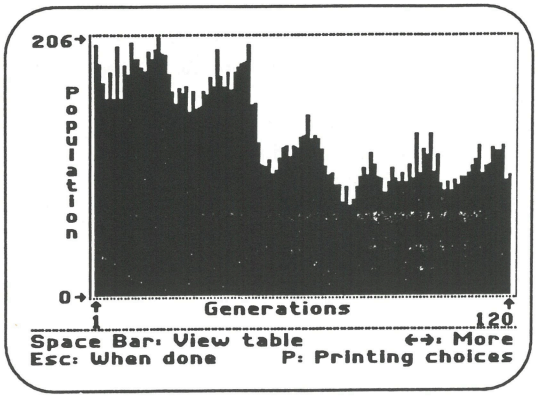
The game screen is a 40 x 27 grid containing 1080 cells. The starting colony consists of 200 dots randomly placed in this grid. This colony represents the first generation of this game. You can move to the next generation by pressing the Space Bar. Press it a few times and notice how each new generation changes and how the population fluctuates.



Now press the A key to watch the behavior of the population. You can adjust the speed (1–5). Eventually, the behavior of living cells will stabilize. This stable state will often include blinking or oscillating patterns and at other times will consist of only "still-life" patterns that do not change. Occasionally, all life will completely disappear. At this point, press the Escape key to exit the game.



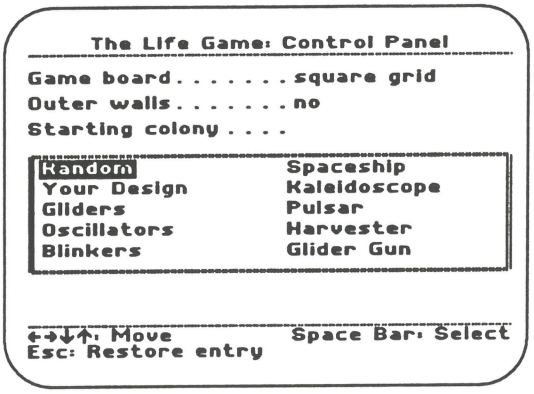
A menu will appear giving you several choices. Returning to the Control Panel allows you to change the current settings or to start a new game with a new starting colony. If you want to see an instant replay of the game you just finished with the same starting colony, choose Option 4 on the menu. If you want to view the history, that is, display a graph of the population, choose the View/Print History option. This option also allows you to print the starting colony or the current colony.



The populations for the last 1000 generations of the game have been saved. Since the graph can only display 120 generations at a time, use the arrow keys to see more of the graph. A table showing the exact population for each generation can be accessed by pressing the Space Bar. The graph or table can also be printed.

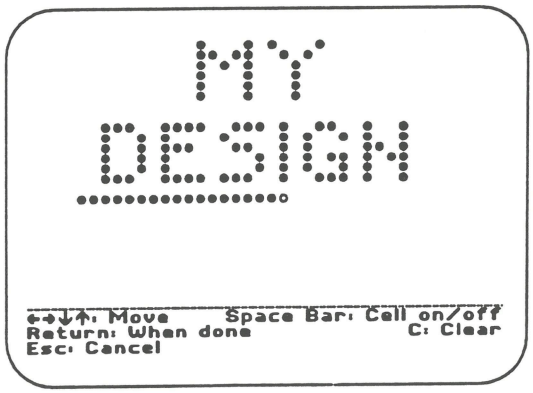
Note: After 9999 generations, the generation counter is reset to 1 and population records are restarted.

Some famous starting colonies and designing your own



Much of the fascination with this program centers around trying to discover new starting colonies that behave in some unusual way or to create interesting patterns. If you select the option to change the starting colony, you will see a menu of choices.

Random: If you have followed along with our previous sample run, you have seen how a colony of 200 randomly placed dots forms a starting colony. The Random mode allows you to determine how many dots (from 1 to 1080) will be randomly placed in the grid to form a starting colony. Using this option is an excellent way to explore.



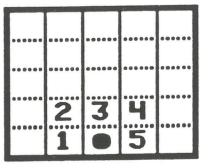
Your Design: With this option, you can create your own starting colony or edit the starting colony from your previous game. If another colony is shown and you want to start from scratch, just press the C key to clear.

Gliders, Oscillators, Blinkers, etc.: Using these options allows you to see a few examples of some noteworthy starting colonies that have been discovered over the years. They were designed to work with the "classic settings." Try each different type and notice the patterns formed and the unique behaviors of different colonies.

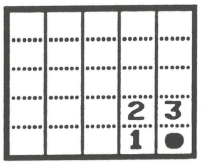
Outer walls—playing Life on the edge!

Conway's original game calls for an infinite grid of square cells to be used as a gameboard. Since this is difficult to obtain, modifications had to be made to simulate the game. An obvious choice was a checkerboard or a piece of graph paper, but these devices were quite limiting and playing the game in this manner was very time-consuming. When the game was adapted to work on a computer, the board size grew to fit the available memory and/or the display device. Often times, the game was played with the understanding that a wall surrounded the gameboard and that cells on the edge of the grid had fewer neighbors. See the diagrams below.

When outer walls are used:



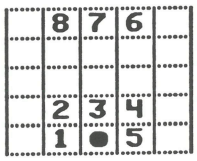
Each edge cell has 5 neighboring cells.

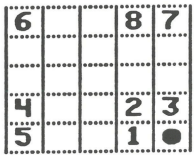


Each corner cell has 3 neighboring cells.

While playing with walls is sometimes limiting and breaks some of Conway's rules, playing on an infinite grid is physically impossible. To overcome this dilemma, the game has been adapted to a variety of surfaces that have a more continuous nature and no walls. Among the surfaces now being used are the sphere, cylinder, torus, and a Mobius strip. When simulating these surfaces on a rectangular computer screen, you see a flattened version with edge cells that have neighborhoods that "wrap around" in order to always have 8 neighbors. This program uses a torus (donut-shaped) model. See the diagrams below.

When outer walls are not used:

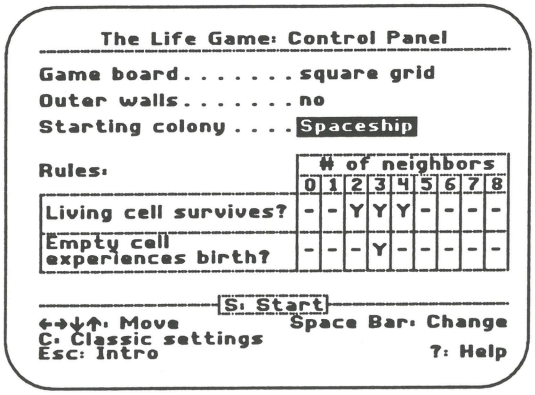




Every cell has 8 neighbors.

Playing The Life Game your way!

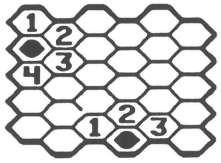
As mentioned previously, Conway experimented with hundreds of different setups for the game before deciding on the grid of square cells and the rules mentioned. Since then, the game of Life has mainly been played in that way. However, the Control Panel in this program provides you with options to change the gameboard and to modify the rules for survival and birth.


The Life Game: Control Panel
Game board . . . . . square grid
Outer walls . . . . . no
Starting colony . . . . Spaceship
Rules:
# of neighbors
0 1 2 3 4 5 6 7 8
Living cell survives? - - Y Y Y - - - -
Empty cell experiences birth? - - - Y - - - - -
S: Start
↔↓↑: Move
Space Bar: Change
C: Classic settings
Esc: Intro
?: Help

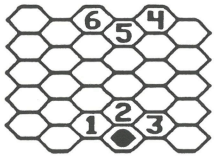
Changing the rules for survival and birth will alter the "classic rule" behavior of colonies. Minor changes can often have dramatic effects. Run the "spaceship" using the classic settings and then run it using this change—living cells with exactly 4 neighbors will also survive. Do you notice any difference in behavior?

Another way to modify the game is to switch to a grid of 760 hexagonal-shaped cells (38 x 20). When using the hexagonal grid, each cell has 6 neighbors.

Each cell has 6 neighboring cells.



With walls, edge cells have fewer neighbors.



Without walls, the neighborhoods "wrap."

Experiment with different settings and see if you can produce some interesting behaviors. You can switch back to the "classic settings" at any time by just pressing the C key as shown on the Control Panel.

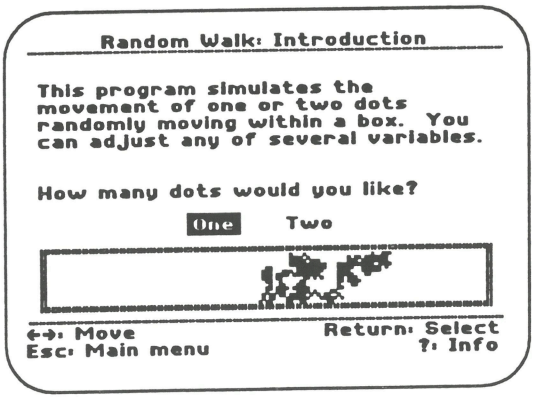
Some suggestions for using this activity in a classroom setting are provided in the "Classroom Ideas" section (pages 37–39).

Random Walk

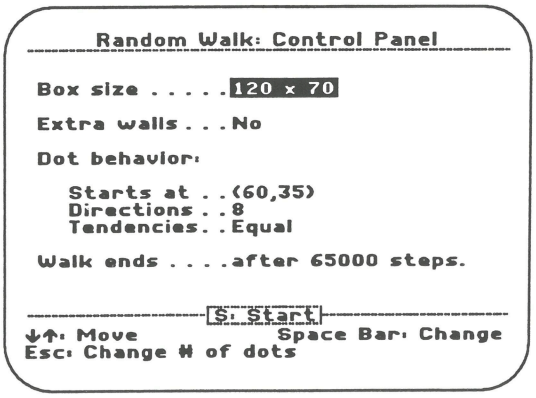
Random Walk is a simulation of one or two dots randomly wandering about within a confined region. Random walks are traditionally topics of interest in many probability courses. They are also used as a method of modeling the behavior of particles, such as microscopic pollen grains, which zigzag their way through the air. More recently, this type of motion (sometimes referred to as Brownian motion) has been used to produce jagged paths that form images known as fractals.

Take a moment and try to imagine the path left by a particle that moves a random distance in a random direction within the confines of a rectangle. As the particle wanders endlessly, its one-dimensional trail gradually starts to fill the two-dimensional region of the rectangle. The image created in your mind must certainly have a dimension greater than one, but since it doesn't completely fill the region, the dimension should be less than two. The concept of a fractional dimension was of great interest to Benoit Mandelbrot. In the early 1970s, Mandelbrot coined the term "fractals" to describe images with dimensions somewhere in between the accepted dimensions of 0 (point), 1 (line), 2 (plane), and 3 (solid). Mandelbrot even used a three-dimensional random walk to generate contours in producing images that became known as fractal landscapes or mountains.

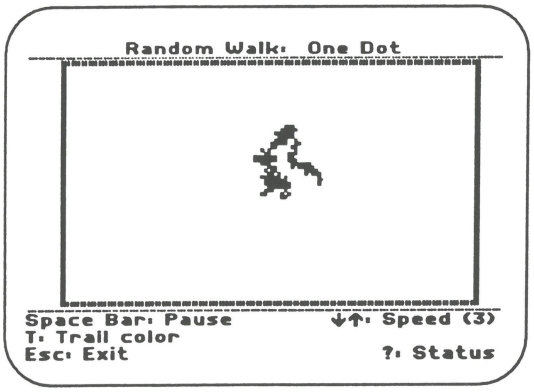
The resolution of the path in **Random Walk** is not fine enough to produce the fractal images sometimes associated with random walks, but it is good enough to effectively demonstrate the behavior of a random dot. Students will be able to witness the unpredictable behavior of random dots and to design simulations based on a random walk. Students will also be able to program the behavior of the dot in an attempt to make the behavior more predictable. The program also includes an editor that allows extra walls or objects to be placed inside the walk area and allows the walk to be ended when certain conditions are completed.


Random Walk: Introduction
This program simulates the movement of one or two dots randomly moving within a box. You can adjust any of several variables.
How many dots would you like?
One
Two
↔: Move
Esc: Main menu
Return: Select
?: Info

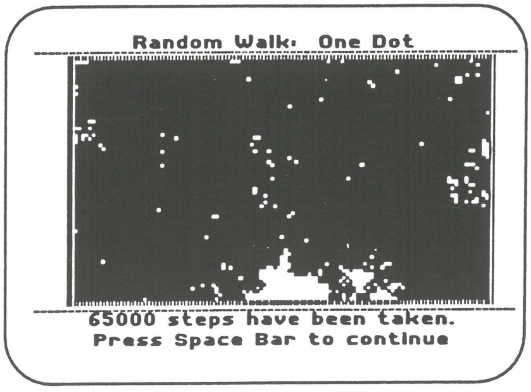
The opening screen introduces the program and shows an example of a random walk in progress. The random walk can be done using either one or two dots. Additional background information is available at this point by pressing the ? key.


Random Walk: Control Panel
Box size . . . . .120 x 70
Extra walls . . .No
Dot behavior:
Starts at . .(60,35)
Directions . .8
Tendencies. .Equal
Walk ends . . . .after 65000 steps.
S: Start
↓↑: Move
Space Bar: Change
Esc: Change # of dots

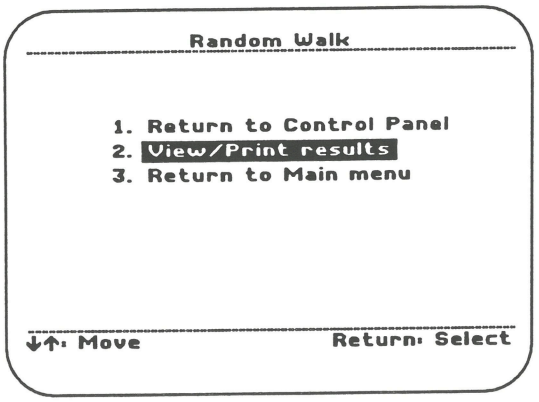
The Control Panel contains the variables that can be changed in designing a random-walk experiment. If all these settings are left as shown, the random walk will consist of a dot randomly moving in any one of eight directions (N, NE, E, SE, S, SW, W, or NW) inside of a box that is a 120 units wide and 70 units tall. The dot is a 1 x 1 unit square. Press the S key to start the walk.



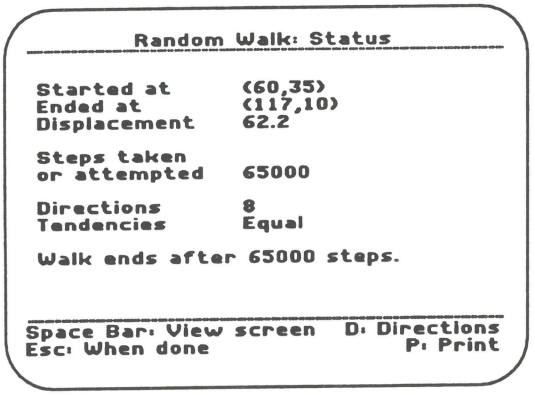
When the walk begins, you will see the dot sitting in the center of the box. To start the dot moving, press the Space Bar. As the dot is wandering, you can regulate the speed by using the arrow keys or pause the walk by pressing the Space Bar again. Pressing the T key will change the color of the trail being left by the dot. The trail will be either white or black. Pressing the ? key shows current settings and allows you to print this information along with the current random-walk screen.



The walk will end if you press the Escape key or if the dot reaches the 65000-step limit imposed, for technical reasons, by the program. Exceeding this limit would have resulted in a much slower moving dot!



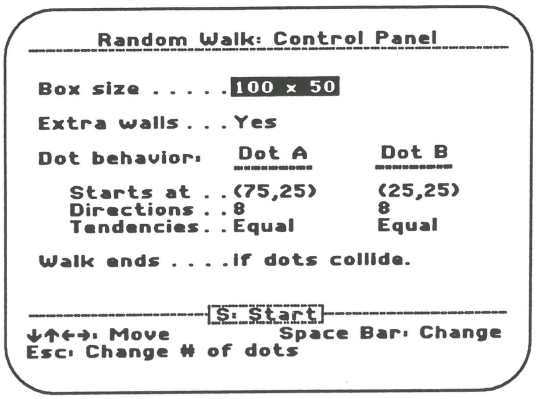
Following each random walk, you are shown a menu of choices, one of which is to view the results of the random walk. When you select the View/Print results option, the screen shown below will appear.

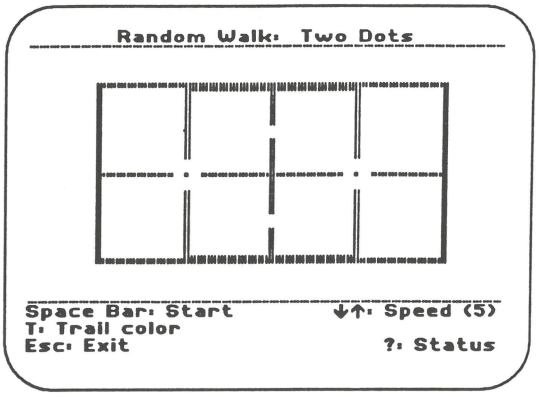


The information collected includes the dot's starting and ending positions, the displacement (distance between these two values), and the number of steps taken.

Note: The number of steps includes all attempts at trying to walk through a wall. When this happens, a clicking sound can be heard. This information, along with a copy of your random-walk screen, can be printed.

Customizing a random walk


Random Walk: Control Panel
Box size . . . . . 100 x 50
Extra walls . . . Yes
Dot behavior: Dot A Dot B
Starts at . . (75,25) (25,25)
Directions . . 8 8
Tendencies . . Equal Equal
Walk ends . . . . If dots collide.
S: Start
↓↑←→: Move Space Bar: Change
Esc: Change # of dots



Box Size: The walk is confined to a box that can be up to 120 units long and 70 units high. If you change the box size, the extra violet or green walls that were added are cleared and the starting position of the dot is reset to the center of the box. The minimum box size is 1 x 1.

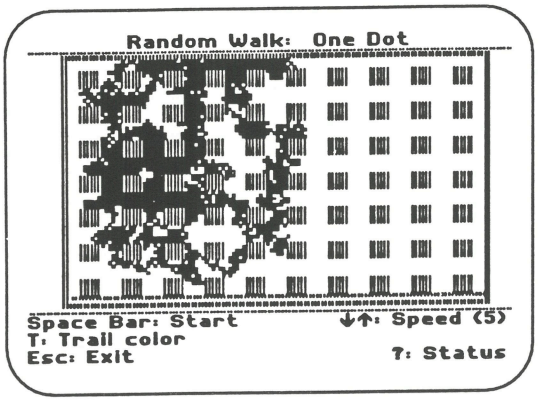
Extra walls: Extra violet walls or obstacles can be added or deleted inside the current box. The editor has a paint brush (white blinking cursor) that can be moved using the arrow keys. To start painting, press the P key and the brush will turn violet. To stop painting, press the P key again and the brush will turn white again. You cannot paint over the random dot. The Maze, Spiral Maze, and City Streets layouts can be used just as they are or they can be edited. If a different sized box is currently selected, it will be automatically changed to 120 x 70 when one of these layouts is selected. Press the C key to clear any extra violet walls added.

Starts at: You can change the starting position of the dot to any location inside the box except on top of a wall. If two dots are being used, they can both occupy the same starting position.

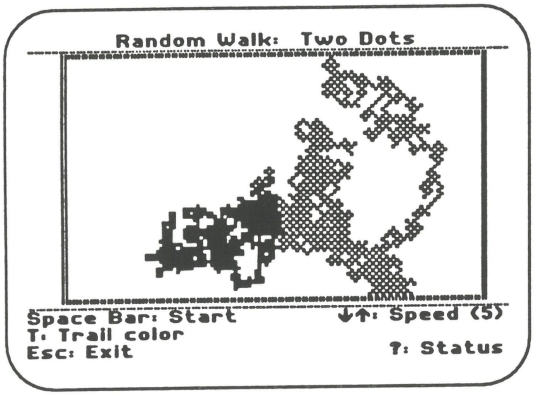
Directions: The dot can move in any combination of the eight compass directions of N, NE, E, SE, S, SW, W, or NW. At least two directions must be selected.

Tendencies: The chances of moving in any of the currently selected directions are initially equal; however, you can vary the tendencies. You must enter all tendencies except the last one, which is automatically calculated so that the total will be 100%.

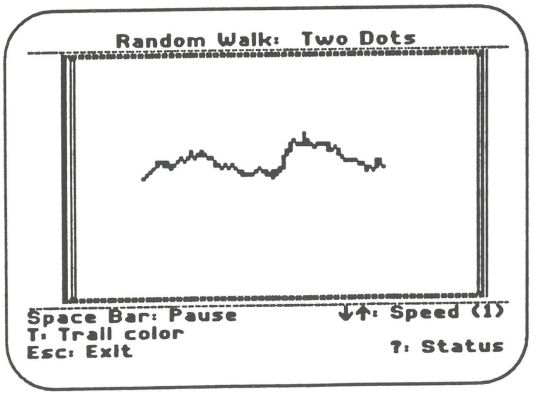
Walk ends: Every walk will end after 65000 steps. You can also set the Control Panel so that the walk will end when a dot hits any wall, when it hits a special green wall, or after a specified number of steps. A special option when using two dots will stop the walk if the dots collide after starting.



This random walk uses the City Streets set of walls. The dot started at the top center and will stop when it touches the green line on the south edge of the city. The green line was added by selecting the option to end the walk if a green wall was hit.



This random walk has two dots. One dot has been programmed so that it can only move in four directions (N, S, E, and W), while the other dot, which is leaving a checkerboard trail in this example, can only move in the other four directions (NE, SE, SW, and NW).



This random walk simulates a type of race between two dots. Both dots started in the center of the screen. The dot on the left side can move in these directions: S, SW, W, NW, and N. The dot on the right can move in these directions: N, NE, E, SE, and S. A green line has been drawn along the left and right sides of the box so that, as soon as one of the dots bumps into the green line, the walk ends.

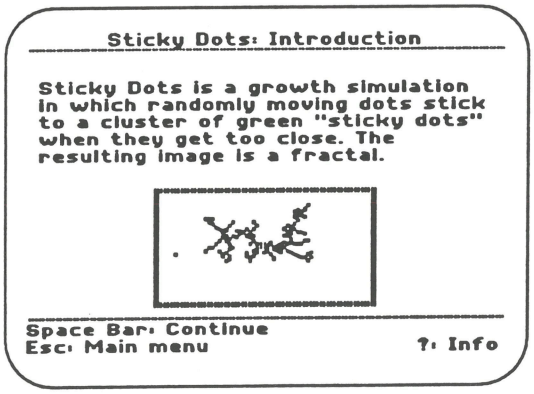
Some suggestions for using this activity in a classroom setting are provided in the "Classroom Ideas" section (pages 40–41).

Sticky Dots

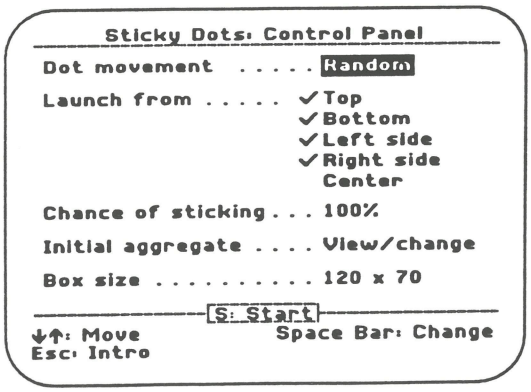
There is an interesting chemistry experiment in which a Petri dish containing a zinc sulfate solution is subjected to an electrochemical process. The anode in this experiment is zinc and the cathode is carbon. When a zinc ion randomly moving about within the solution wanders too close to the carbon cathode, it is attracted and sticks. After a few minutes, lacy branches of zinc deposits become quite visible. Within a short time, the branches extend almost to the outer edges of the Petri dish. The result of this process is a fractal structure.

The process outlined in this experiment is easy to describe but is not a process that is well understood by researchers. But the process, called diffusion-limited aggregation (DLA), is quite simple to simulate with the help of a computer. Imagine a stationary cluster of dots positioned in the middle of your computer screen. This cluster is known as the initial aggregate. A dot is launched from the edge of the screen and allowed to wander randomly about the screen. Sooner or later the dot wanders to a position right beside the cluster. When this happens, the dot sticks and becomes part of the cluster. A new dot is then launched and the process continues. Eventually, the structure grows to fill the screen. A fractal has been created!

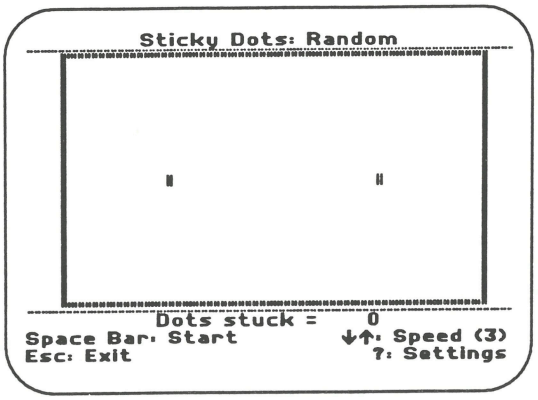
Sticky Dots allows students the freedom to design many different types of environments for creating fractal images using the process of aggregation. Two different types of dot movement—random and linear—can be used, and the size, shape, and position of the initial aggregate can be changed. An interesting additional feature is the ability to change the probability of the dot sticking to the aggregate.


Sticky Dots: Introduction
Sticky Dots is a growth simulation in which randomly moving dots stick to a cluster of green "sticky dots" when they get too close. The resulting image is a fractal.
Space Bar: Continue
Esc: Main menu
?: Info

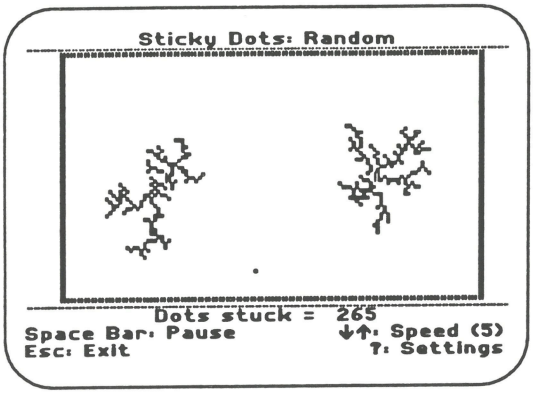
The first screen introduces the program and also shows a running example of a "sticky-dot" simulation. Some background information is available by pressing the ? key.


Sticky Dots: Control Panel
Dot movement . . . . . Random
Launch from . . . . . ✓Top
✓Bottom
✓Left side
✓Right side
Center
Chance of sticking . . . 100%
Initial aggregate . . . . View/change
Box size . . . . . 120 x 70
S: Start
↓↑: Move
Space Bar: Change
Esc: Intro

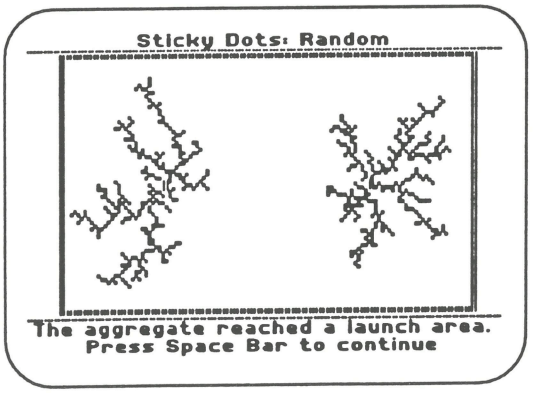
The Control Panel for **Sticky Dots** allows you to change any of the variable settings shown. If all of these settings are left as is, the simulation will be similar to the one shown in the next screen. To go to the simulation screen, press the S key.



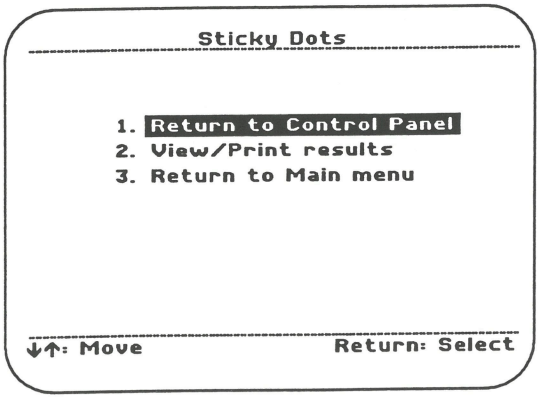
This is the largest box possible, and the initial aggregate is two small areas of green. The dots will be launched from random locations along the top, bottom, left, and right sides of the box. The dot will wander, using a random-walk motion, and a count of the number of dots caught by the green "sticky stuff" will be shown at the bottom of the screen. You can start the process by pressing the Space Bar.



At first, the process can take quite a long time; but as the structure increases in size, the amount of time for a dot to stick will decrease. You can use the arrow keys to adjust the speed. If you are using an Apple IIgs, be sure that your system speed is on fast. At any time during the run, you can review your settings by pressing the ? key. This option also allows you to print the current image along with the settings.

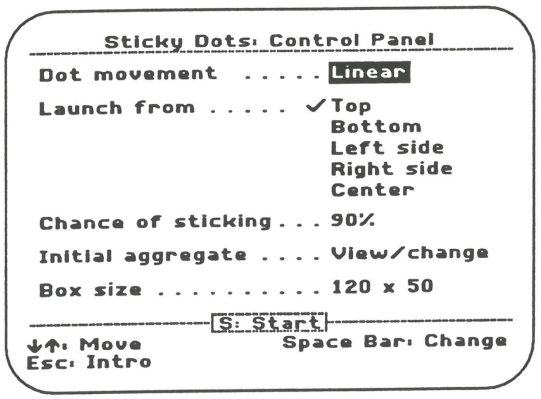


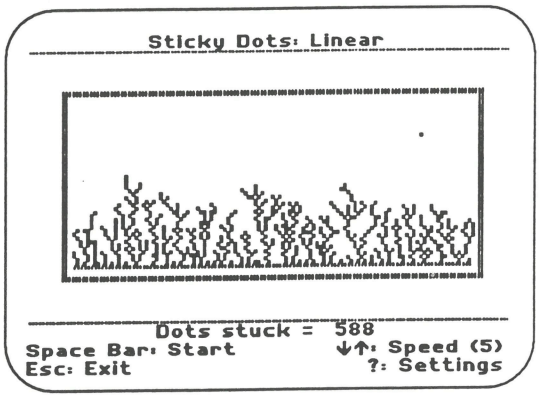
The simulation ends when the fractal structure reaches one of the launch areas. You can also end the run before this happens by pressing the Escape key.



Following the run, a menu of choices will appear. Returning to the Control Panel will let you rerun the experiment or change settings for a different experiment. If you choose to view the results, you will see the settings used and the image that was produced. If printing is allowed, this information can be printed.

Understanding the Control Panel


Sticky Dots: Control Panel
Dot movement . . . . . Linear
Launch from . . . . . ✓Top
Bottom
Left side
Right side
Center
Chance of sticking . . . 90%
Initial aggregate . . . . View/change
Box size . . . . . 120 x 50
S: Start
↓↑: Move
Space Bar: Change
Esc: Intro



Dot movement: The dot is always launched from a random location in one of the currently active launch zones. Once launched, the movement can be either random or linear. When linear is selected, the dot will travel in a straight line to the opposite side of the box. If launched from the center, it will travel toward one of the outer edges.

Launch from: Any combination of the launch zones can be selected. The center launch zone is a 2 x 2 area in the center of your box.

Chance of sticking: The probability of the dot sticking to the aggregate when it lands next to it can be set anywhere in the range of 1% to 100%. When the dot is in a position to stick but does not, a tone can be heard.

Initial aggregate: The default settings place two small clusters near the center of the box. You can use the editor to clear these and paint in your own green "sticky stuff." Press the C key to clear the current aggregate. Press the P key to raise and lower the paint brush. You will not be allowed to paint in the launch areas along the edges of the box or in the center. Attempting to paint over the center area will cause a beeping sound to occur. Press Return when done or press Escape to cancel.

Box size: The maximum size box is 120 x 70. The minimum size box is 20 x 20. When you change the box size, any initial aggregate that you may have added will be erased and replaced with the two default dots.

Some suggestions for using this activity in a classroom setting are provided in the "Classroom Ideas" section (page 42).

The Chaos Game

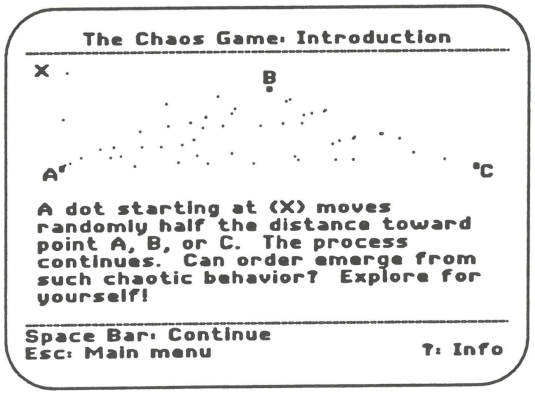
In 1988 Michael Barnsley, from the Georgia Institute of Technology, published an article describing a rather unique process for plotting points to produce certain types of images. The process, which he called the Chaos game, required a piece of paper, pencil, a ruler, and a regular six-sided die. The steps in playing the game are as follows:

Draw any triangle and label the vertices A, B, and C and then locate a starting point anywhere on the paper. To determine where the next point will be plotted, roll the die, check the value, and apply the appropriate rule:

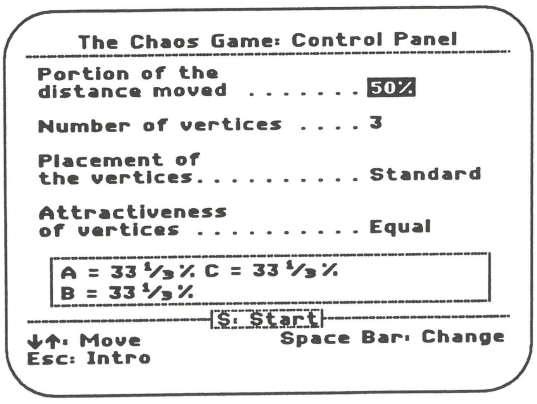
- If a 1 or 2 is rolled, plot the next point half way between the current point and vertex A of your triangle.
- If a 3 or 4 is rolled, plot the next point half way between the current point and vertex B of your triangle.
- If a 5 or 6 is rolled, plot the next point half way between the current point and vertex C of your triangle.

Roll the die again and repeat the process using the point just plotted as your current point. Continue the game for several rounds.

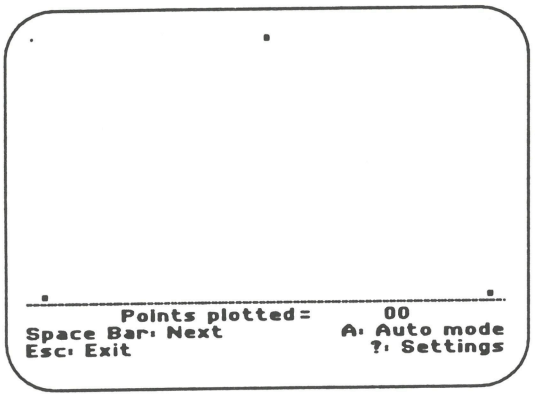
After a few rounds, stop and look at the arrangement of the points plotted. Can you predict what it will look like after several hundred points have been plotted? Most players would probably be convinced that the resulting image will be a well-scattered set of random dots. Some players might even be tempted not to use the ruler and die and merely plot several dots at random. But patient players will eventually notice that a surprising amount of order does exist in this rather random process. This game has caught the attention of many mathematicians who wonder if similar types of iterative processes will have similar results.


The Chaos Game: Introduction
X
B
A
C
A dot starting at (X) moves randomly half the distance toward point A, B, or C. The process continues. Can order emerge from such chaotic behavior? Explore for yourself!
Space Bar: Continue
Esc: Main menu
?: Info

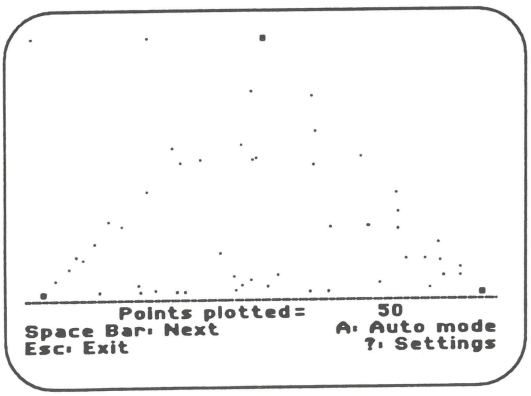
The opening screen introduces the game and shows the results of plotting a few points. Some additional background information about the game is also available by pressing the ? key.


The Chaos Game: Control Panel
Portion of the distance moved 50%
Number of vertices 3
Placement of the vertices. . . . . Standard
Attractiveness of vertices . . . . . Equal
A = 33 1/3% C = 33 1/3%
B = 33 1/3%
S: Start
↓↑: Move
Space Bar: Change
Esc: Intro

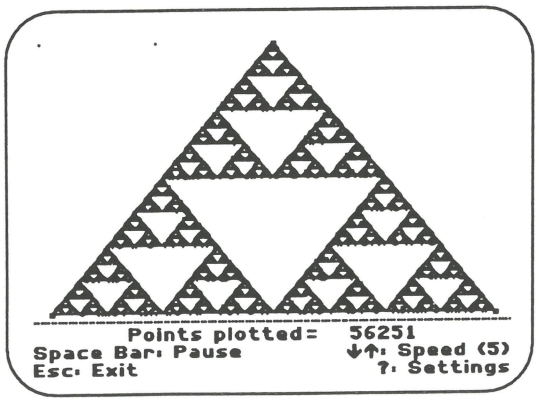
The next screen is the Control Panel. Any of the variables shown can be changed in order to modify the game. The initial Control Panel settings are for the traditional game, using a triangle and moving half the distance toward one of the vertices on each move. If these settings are kept, the game will be played as shown in the following screens.



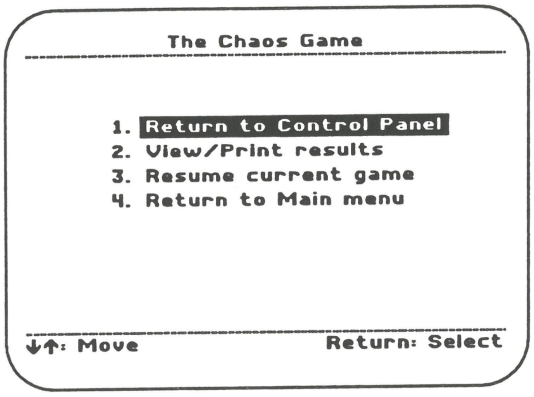
The initial game screen will show the location of the three vertices of a triangle and a starting point in the upper left corner. New points can be plotted one at a time by pressing the Space Bar or automatically by pressing the A key.



Slowly plot a few points by pressing the Space Bar and noting the location of each new point. Notice how it is placed midway between the last point plotted and one of the three vertices. The color of the points—violet, green, or white—has no significance. Continue until about 50 points have been plotted. The image seems rather random and many first-time viewers will predict that things will not improve. Now shift the process into the auto mode by pressing the A key.

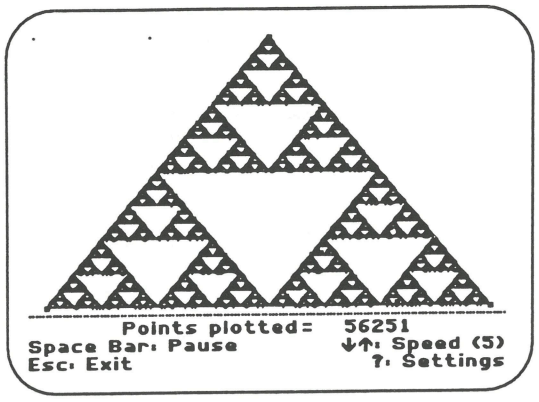


After a few seconds, a definite pattern will begin to emerge. The speed of plotting the new points can be regulated by using the arrow keys. There are five speeds available. At any time during the game, you can review the settings by pressing the ? key. While viewing the settings, you can also print the current image and settings. The game will continue until you press the Escape key. The counter for the number of points plotted will reset to 1 after 6,553,600 points are plotted.



When you exit the game, a menu of choices will appear. If you choose to view the results and printing is allowed, you will be able to print a copy of the image generated and the settings.

Taking a closer look at the results

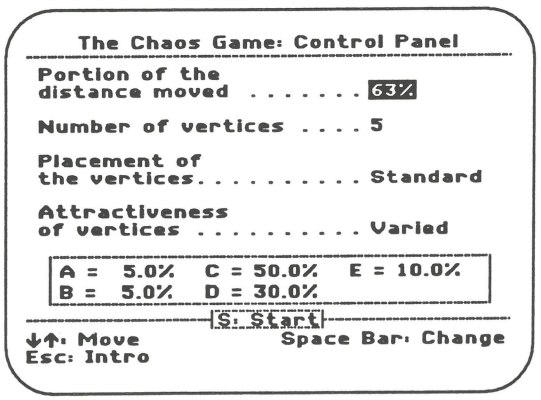


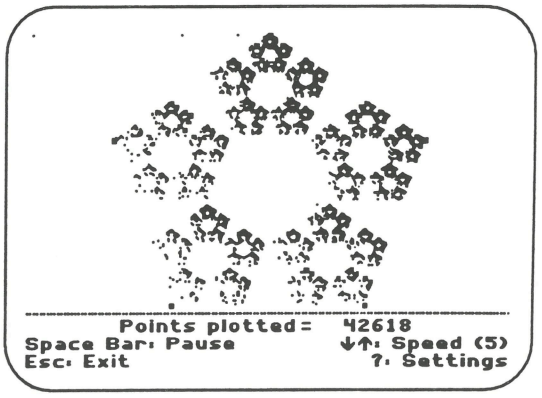
The traditional version of the game generates a rather porous triangle. When witnessing the process for the first time, there is usually a sense of disbelief that such an orderly arrangement of points can emerge from such a random process. To many mathematicians, the image was not only orderly, but a very familiar image. In the early 1900s, Polish mathematician Waclaw Sierpinski suggested that a most interesting shape could be made from a triangle. His idea was to take a triangle and cut out a center triangle formed by connecting the midpoints of the sides of the triangle. The process was repeated with the three remaining triangles and then continued indefinitely with the triangles that remained after each step or iteration. The resulting figure is referred to as a Sierpinski Triangle.

The Sierpinski Triangle is an image known as a deterministic fractal with a dimension of about 1.58. (See page 16 for a brief explanation of fractional dimensions.) The image clearly exhibits a property of fractals known as "self-similarity." This means that smaller parts of the image look similar to the whole image. When the smaller parts look exactly like the larger image, as in the Sierpinski Triangle, the image is said to be "strictly self-similar."

The process used in producing the image is an important technique known as iteration. Iteration occurs when some function is applied to an initial input and the resulting output is then used as the input when applying the function again. In **The Chaos Game**, the starting point is the initial input. The function is the rule for rolling the dice and determining where to locate the new point. The output is the new point plotted. So each new output point determined by the function becomes the input point the next time the function is used.

Making changes to The Chaos Game


The Chaos Game: Control Panel
Portion of the distance moved 63%
Number of vertices 5
Placement of the vertices. . . . . Standard
Attractiveness of vertices . . . . . Varied
A = 5.0% C = 50.0% E = 10.0%
B = 5.0% D = 30.0%
S: Start
↓↑: Move
Esc: Intro
Space Bar: Change



Portion of the distance moved: This option refers to the portion of the distance to move from the last point plotted toward the vertex randomly selected. The portion can be adjusted within the range of 10% to 90%.

Number of vertices: The number of vertices can be set to 2, 3, 4, 5, or 6.

Placement of the vertices: An editor is available to reposition any of the vertices or the starting point. When the number of vertices is changed, they are placed in a standard position that closely resembles a regular polygon. To change the position of any point, press the N key until the point to be moved has been selected. The vertex point currently selected and its attractiveness will be indicated in the box below the line. Then use the arrow keys to move the point to the desired location (or ⌘-arrows to move, using longer steps). When you have completed positioning all the points, press the Return key. Pressing the Escape key at any time during the session will cancel your work and return the points to their original positions. The points can also be returned to the standard (default) positions by pressing the R key.

Attractiveness of vertices: When first entering the Control Panel or when the number of vertices is changed, the attractiveness of the vertices is equal. This means that the chance of the point moving toward any one vertex is the same as it is for any other vertex. However, these chances can be varied so that the point is attracted more toward some vertices than others. As the percentages are entered, the current total will be displayed. This total must add up to 100%. To simplify this task, the last value is automatically calculated and displayed so that the total is 100%.

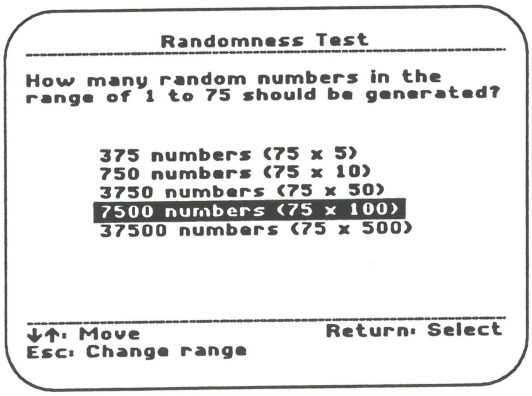
Some suggestions for using this activity in a classroom setting are provided in the "Classroom Ideas" section (pages 43–44).

Randomness Test

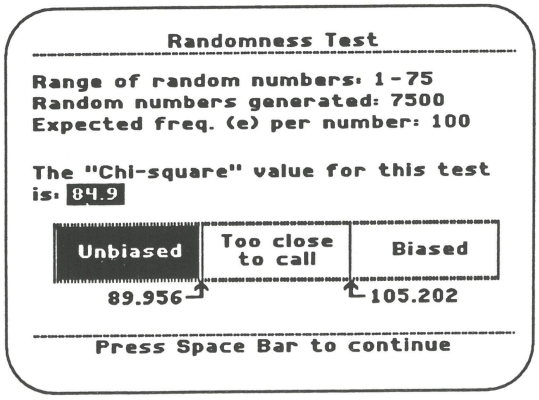
The five programs in *Exploring Chaos* all rely to some degree on the use of a pseudo random-number generator. The AppleSoft RND function (random-number generator) is not used by these programs. Instead, a faster generator has been incorporated in order to improve program performance. Regardless of the routine used in this or any other computer experiment that uses random numbers, the randomness of the generator is a critical factor in ensuring reliable results. For this reason, a randomness test has been included.

The range of random numbers used in each program varies. For example, a dot in **Random Walk** might move randomly in one of eight directions, while a dot in **Sticky Dots** might have only a 1% chance of sticking. Also, the number of times that a random routine is used in each program varies. In **Mosaic Madness**, one might randomize the rules only once, but in **The Chaos Game**, one might allow a quarter million dots to be plotted using a random-placement routine. For these reasons, some flexibility has been designed into the test.

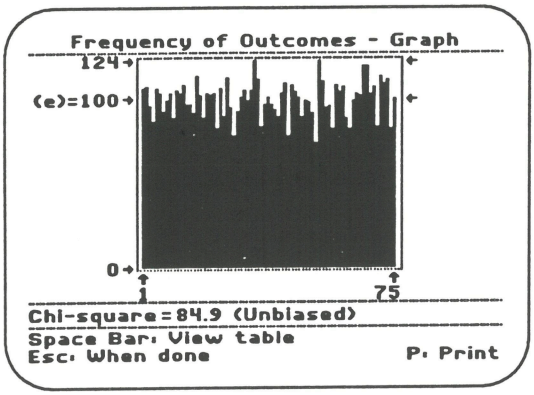
The randomness test used in *Exploring Chaos* is the popular Chi-square test. Once calculated, our Chi-square value is compared to values found in a probability table of Chi-squares. This comparison indicates that our random-number generator is either unbiased, too close to call, or biased. More information about the test and the tables can be found in most statistics textbooks.


Randomness Test
How many random numbers in the range of 1 to 75 should be generated?
375 numbers (75 x 5)
750 numbers (75 x 10)
3750 numbers (75 x 50)
7500 numbers (75 x 100)
37500 numbers (75 x 500)
↓↑: Move
Return: Select
Esc: Change range

This test allows you to specify a random number range between 1 and N ($2 \le N \le 100$) and also to decide how many random numbers should be generated within the specified range. This quantity is always a multiple of N.



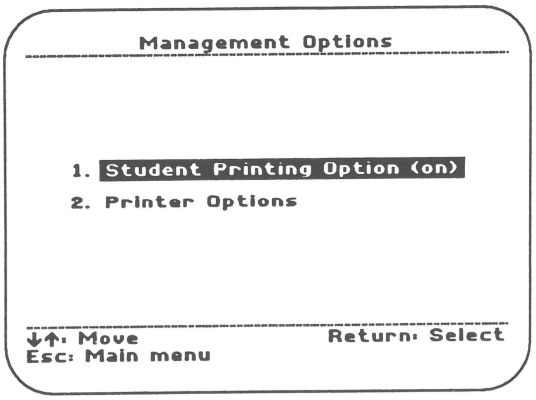
After the test has been completed, the Chi-square value is displayed along with an evaluation of the random-number generator. The two numbers displayed at the bottom of the screen are taken from the Chi-square probability table. If our generator is really fair, the chance of our value exceeding the first value is 10%, and the chance of exceeding the second value is only 1%.



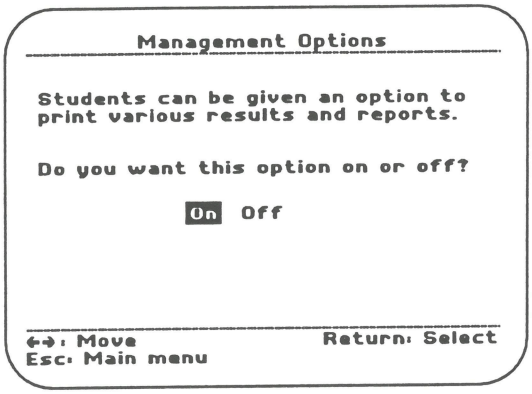
A frequency of outcomes graph and table are also available to view the actual performance of our random-number generator in this test. This information can also be printed.

Management Options

Accessing Management Options

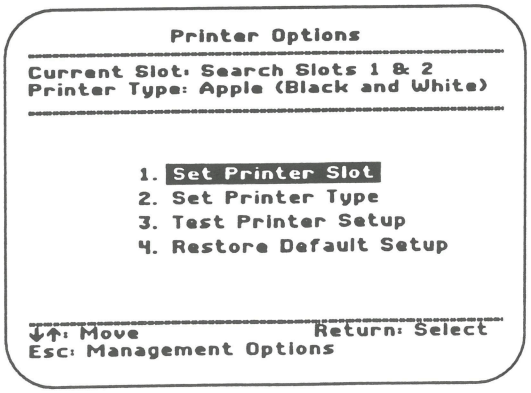


The Management Options can be accessed from the Main menu by typing Control-A (hold down the Control key and press the A key). The menu shown here will then appear.


Management Options
Students can be given an option to print various results and reports.
Do you want this option on or off?
On Off
←→: Move
Return: Select
Esc: Main menu

Student Printing Option

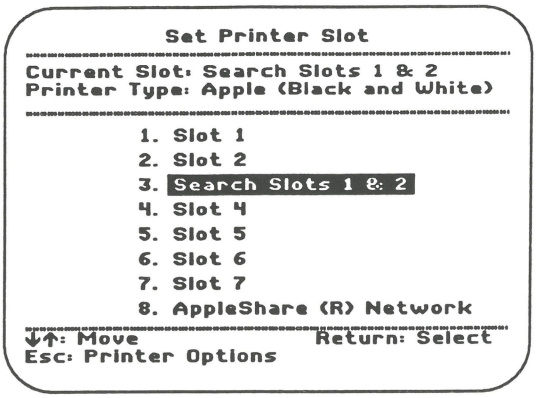
Program results, as well as Control Panel settings, can be printed. The Student Printing option allows you to regulate printing. When turned off, students will not see the "P: Print" command. The current status (On or Off) of this setting is reflected on the Management Options menu.



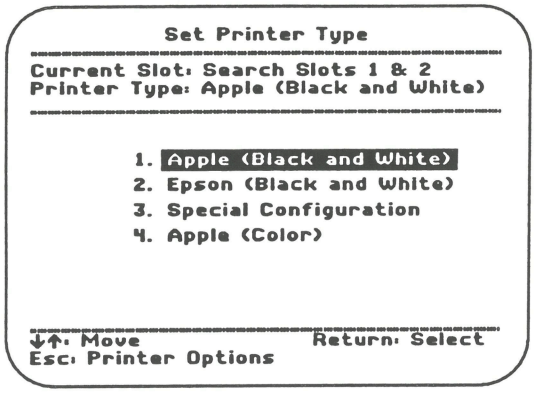
Printer Options

When you choose Printer Options, you will see this menu. The current printer settings are shown at the top of the screen. Using this menu, you will be able to:

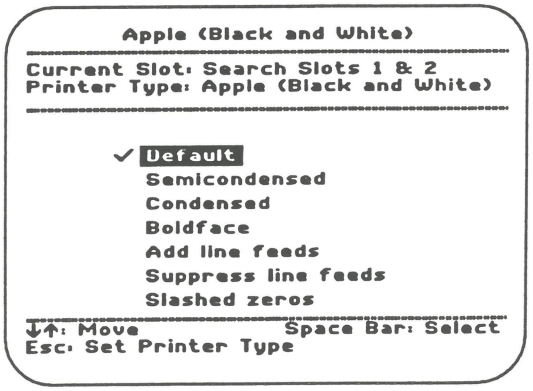
- set the printer slot;
- set the printer type;
- test the printer setup;
- restore the default setup.



Set Printer Slot allows you to specify the slot in which your printer interface card is located. If you are printing across an AppleShare® Network, choose Option 8. This setting will automatically select the correct printer slot for each computer on the network.

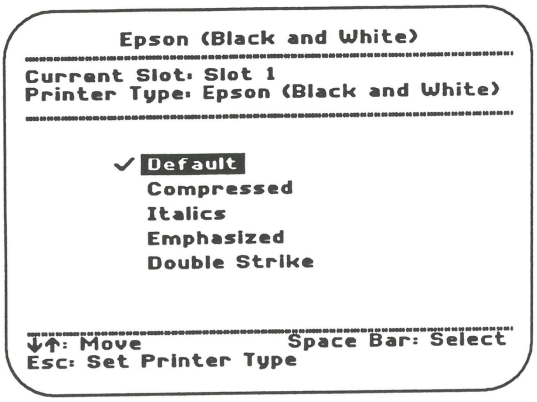


Set Printer Type allows you to select the type of printer you are using: either an Apple (Black and White), an Epson (Black and White), a special configuration, or an Apple (Color).



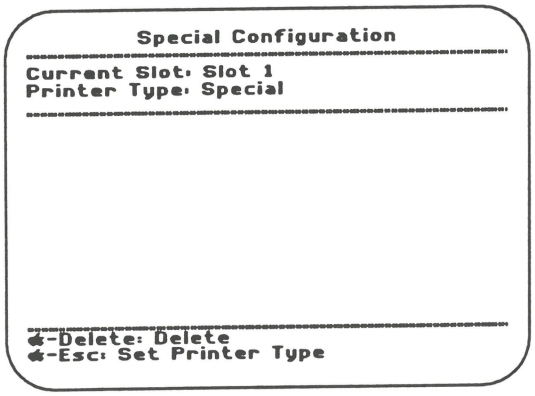
Selecting Apple (Black and White or Color) allows you to choose one or more of the options shown here. However, you are not able to choose Semicondensed and Condensed or Add line feeds and Suppress line feeds at the same time.

Choosing Default clears all other customized options or special commands. To ensure that the printer also clears these commands, turn your printer off for a few seconds and then turn it back on before printing.



Selecting Epson (Black and White) allows you to choose one or more of the options shown here. However, you are not able to choose Compressed and Emphasized at the same time.

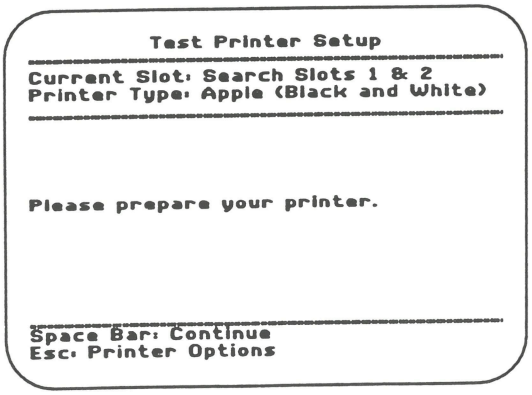
Choosing Default clears all other customized options or special commands. To ensure that the printer also clears these commands, turn your printer off for a few seconds and then turn it back on before printing.



Selecting Special Configuration allows you to enter commands that enable certain types of printers to operate. These special commands are listed in the manufacturer's printer manual.

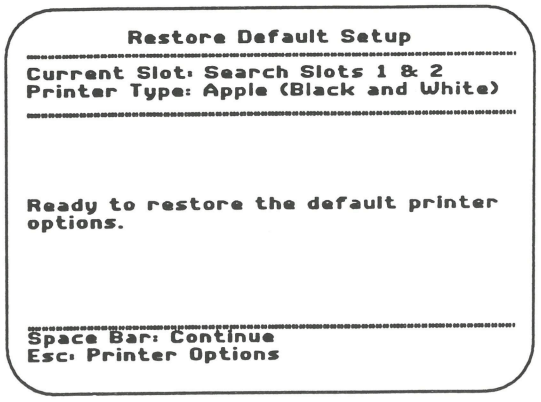
To enter special commands, type the exact characters required. When finished, type ⌘-Esc to end.

Do not set up the printer to use a proportional font. This setting will cause reports to be formatted incorrectly.


Test Printer Setup
Current Slot: Search Slots 1 & 2
Printer Type: Apple (Black and White)
Please prepare your printer.
Space Bar: Continue
Esc: Printer Options

Test Printer Setup prints out all of the keyboard characters. If these characters are not printed correctly, check the settings on your printer and interface card, check to see whether your printer has been connected correctly, or look at your interface card manual for special commands.

Note: To ensure that previous printer commands are cleared, turn the printer off, wait a few seconds, and then turn it back on before performing this test.


Restore Default Setup
Current Slot: Search Slots 1 & 2
Printer Type: Apple (Black and White)
Ready to restore the default printer options.
Space Bar: Continue
Esc: Printer Options

Restore Default Setup returns all printer settings to their original state. The original printer setup provides a search of Slots 1 and 2 for an Apple printer.

All changes made to the Printer Options are saved on the disk and are permanent until you use the Printer Options again to change the printer settings.

Classroom Ideas

Mosaic Madness

Mosaic Madness is an example of a linear or one-dimensional cellular automaton. Before using the program, you may want students to look up the definition of an automaton (singular) or automata (plural). If you have reference materials dealing with the topic, you could also have students read about uses of cellular automata in modeling various systems or processes.

Perhaps the easiest way to understand how the program works is to simply run it and observe the results. Each new row of tiles is generated by observing a set of coloring rules. The rules are based on the sum of the color values for the parent tiles above each new tile generated.

Some Questions Worth Investigating

1. Stephen Wolfram has done extensive research on these models. His findings indicate that there are four basic types of patterns that evolve. Can you discover what these are? Do some types seem to occur more often than others?

   (Answer: Through his experiments, Wolfram made the following observations: 1) starting patterns die out about 50% of the time; 2) patterns degenerate into homogeneous states (vertical stripes) about 25% of the time; 3) patterns grow at a fixed rate about 16% of the time; and, 4) patterns grow and contract irregularly about 9% of the time.)

2. What role is played by the initial conditions (the top row) in determining the patterns produced? Generate several different mosaics until you find one that has a distinctive pattern. Retain the rules and make changes only in the top row. Try some very minor changes and also some major changes.

3. What role is played by the coloring rules in determining the patterns produced? Generate several mosaics in which the same top row is always used, but where the rules change each time. Try some rule changes in which very minor changes are made and others that have major changes.

4. Can you generate the Sierpinski Triangle (see page 27) using the offset rows? (Set each tile value in the top row to 0 except for the center tile. Set it to 3. Set all the new tile values in the rules to 0 except for when the sum of the two parents is 3. Set that value to 3.)

5. Can you produce a checkerboard design?

Follow-up or Extension Activities

1. Have students print interesting patterns they have discovered and post them on the bulletin board.

The Life Game

The game of Life is a classic computer-science activity. It was first introduced in the October 1970 issue of *Scientific American* (pages 120–123). You may want to read the article yourself or share it with your students. That article created so much interest among readers that another column in the February 1971 issue (pages 112–117) was also devoted to the game. In the first article, Conway challenged someone to disprove his conjecture that a finite colony could not grow without limit and offered a $50 reward to the first one who could do so. In the February issue, a group of scientists at M.I.T. disproved Conway's conjecture through their creation of a finite colony known as the "Glider Gun."

Review the rules of the game with your class. Students might be a little confused at first, but be sure to emphasize how a cell survives or dies, based on the number of neighbors. Also review how new cells are born when the number of neighbors surrounding an empty cell is "just right."

Stick with the classic settings to start with and have your students try the different starting colonies provided. Encourage students to design their own starting colony in order to become familiar with the editor provided. An interesting activity is to have students draw their name and see how many generations it can go before disappearing or stabilizing.

Some Questions Worth Investigating

1. Is the initial population of a random-starting colony related to the number of generations it takes to disappear or stabilize? If so, what seems to be an ideal starting population to produce the longest life before stabilizing?

2. A tetromino is a colony of four neighboring cells. There are five basic ways of building tetronimoes. Can you design and test each type? How about the family of twelve basic pentominoes (five neighboring cells)?

3. An "oscillator" is a colony that returns to its original state after several generations. The simplest of these is called a "blinker" and alternates between two different states. The number of generations required for a colony to return to its original state is referred to as the "period." A blinker has a period of two. Can you find an oscillator with a period of three? four? five?

4. Some colonies like the "Pulsar" produce very symmetrical designs. Can you discover others?

5. Can minor changes in the starting colony drastically change its behavior? Run the Spaceship a few times and become familiar with its behavior. Then add or delete one cell and observe the behavior. Try this with some other starting colonies.

6. Can you discover a "Garden of Eden" colony that can have no predecessor, that is, one that could only exist as a starting colony.

7. Minor changes in the rules can result in very different behaviors. Start with the classic settings and run the Pulsar. Notice the behavior. Now change the rule for survival so that cells with 4 neighbors also survive. Run the Pulsar again and compare the behaviors. We have labeled these as the "city builder" rules.

8. When Conway was designing the game, he experimented with many different grids and rules. One grid was the hexagonal grid, where each cell has six neighbors. **The Life Game** program allows you to design games using this grid. When using a hexagonal grid, some rules produce very symmetrical patterns. To illustrate, run the Single Dot starting colony with these rules: cells with 2 or 3 neighbors survive and empty cells with 1 neighbor experience birth. Can you find other starting colonies or rules that have similar results?

Questions Worth Researching

1. Conway's original game of Life calls for an infinite grid of squares to be used as the gameboard. For obvious reasons, that would be a difficult game to model! Our version uses a flattened torus (donut-shaped) for a gameboard. Are there other surfaces that have been adapted for playing the game?

2. What was Conway trying to demonstrate when he developed his game?

3. **The Life Game** is an example of a two-dimensional cellular automaton. What is meant by a cellular automaton and why were they developed?

4. Many unique colonies have been discovered that exhibit unusual behaviors. Among those that have been catalogued are the Frog, Glider, Pinwheel, Spaceship, Powder Keg, and the Snake. Can you find out more about these colonies and experiment with them?

Follow-up or Extension Activities

1. The game of Life was of particular interest to computer scientists because it was well suited as a computer simulation. New players who do not have access to a computer should use large sheets of graph paper or checkerboards. Try to simulate a game starting with a small colony, like a single glider, using a checkerboard and checkers.

2. Print out copies of interesting colonies students discover and post them on the bulletin board. Have the students give their colonies names.

Random Walk

Random walks are often used as the basis for problems in probability courses and to model the behavior of certain types of particles such as metallic ions or grains of pollen moving through the air. Discuss the meaning of a random walk and try to think of examples in life that exhibit the behavior of a random walk.

Some Questions Worth Investigating

1. How well can you predict the outcome of a random walk? Starting with a single dot in the center of a box, predict how many steps will be taken before the dot hits a wall. Set the program so the walk will end if any wall is hit. Make your prediction and run the program. Were you close? Revise your prediction and run it again. Repeat this ten times. What was the average number of steps taken in each run? How does this compare to the findings of others?

2. The chances of a dot being found far away from the starting position increases as the number of steps increases. Is there a relationship between the number of steps taken and the displacement from the starting position? Start with a random walk on a straight line in which the dot can only move east or west. Gather displacement data from several runs that end after exactly 10 steps, 50 steps, 100 steps, etc. Can you find a relationship in a two-dimensional random walk? Repeat the experiment with the dot able to move in any of the eight compass directions.

3. Can you invent any games using the random-walk concept? How about a race between two dots or some version of "Capture the Flag?"

Questions Worth Researching

1. **Random Walk** is an example of Brownian motion. What is Brownian motion and what are some of the things that exhibit this type of motion?

2. Would it be possible for a randomly moving point to hit all the points inside a region, given enough time? How would one describe the one-dimensional path left by a point as it gradually fills the two-dimensional space?

Follow-up or Extension Activities

1. **Random Walk** can be adapted to model many different types of problems studied in probability.
2. Print copies of interesting runs or games invented and post them on the bulletin board.

Sticky Dots

The **Sticky Dots** program is based on the concept of a random walk. If you have not already done so, demonstrate the **Random Walk** program or have your students run it.

Explain what the initial aggregate is and how dots stick and become part of this structure. Demonstrating the program at a low speed will help students understand what is happening. After watching the process for a few minutes, students may start to wonder what would happen if the initial aggregate were changed or a different type of movement were used. Let students begin exploring with the program. If time is a factor, you may want to suggest that students reduce the size of the box used.

Some Questions Worth Investigating

1. This program has two different types of dot movement, random and linear. What is the difference between these movements? Are the resulting fractal images formed by these two types of movements different? Why?
2. Does limiting the launch areas change the shape of the fractals formed?
3. Most models of this type assume that the chances of a dot sticking are 100% when it lands beside an aggregate. What happens to the resulting fractal when the chances of sticking are lowered?

Questions Worth Researching

1. Diffusion-limited aggregation and ballistic aggregation are processes that are often found in nature. Define these two processes and find some examples of each.
2. What if several particles were simultaneously released in this program? What if the aggregate structure itself were mobile? Do you think the resulting fractal images would be much different?

Follow-up or Extension Activities

1. The chemistry experiment mentioned on page 21 is an actual example of aggregation that can be demonstrated during a class period. A description of the experiment can be found on pages 399–405 in the book *Fractals for the Classroom—Part One* by Peitgen, Jurgens, and Saupe (New York, Springer-Verlag, 1992).
2. Many different fractal images can be produced. If a printer is available, print the outcomes and post them on the bulletin board.

The Chaos Game

Perhaps the best way to introduce **The Chaos Game** is to have students play it by using Worksheet 3 (page 48). Have students plot about 20 points and then write down their predictions as to what kind of pattern, if any, will emerge. If time permits, make an overhead transparency of the worksheet and pass it around for students to place on top of their worksheets and mark the points they found. In this way, a large number of points can be compiled on a single source and the results easily displayed. Did a pattern emerge?

An alternate way of introducing the activity is by using a large-screen projection system and demonstrating the program through its default settings. First, go over the rules of the game with the class and then plot one point at time, using the Space Bar to generate the next point. As each point appears, note its location, using a pointer, and follow the progress of the point for several iterations. After 40 or 50 points have been plotted, have each student write down what they think the mass of points will look like after 10,000 have been plotted. If you like, have students share their predictions. Then put the program in Auto mode by pressing the A key. Did anyone make the right prediction?

At this point, students will probably be curious to know how this happened. Some will want to move the starting point and see if that will change the outcome. Others will be curious to know what would happen if you moved less than or more than 50% of the distance. This is probably a good time to let students begin exploring on their own. You may even want to generate a list of questions to be researched and then assign small groups the task of finding the answers.

Some Questions Worth Investigating

1. Does the position of the starting point have any effect on the outcome? Does placing it outside, inside, or on the figure make any difference?

2. Occasionally, within the image, a few stray dots will appear that do not seem to belong. How can you explain this?

3. What happens if the size or shape of the original triangle is changed? Try several with different sizes and shapes and find out.

4. What happens when the portion of distance moved toward each vertex is changed? Compare the results from several runs. If a printer is available, print the results and arrange the images according to the portion of the distances. Does there seem to be any relationship between the amount of the blank area and the percentage of the distance?

5. What happens when the attractiveness of the vertices becomes biased? Compare the results from several runs.

6. Does changing the number of vertices produce images that you would expect?

Questions Worth Researching

1. Will the same point ever be plotted twice?

2. What is meant by "self-similarity" and is it evident in the images produced using **The Chaos Game**?

3. What is the area of the Sierpinski Triangle?

Follow-up or Extension Activities

1. Another method of producing the Sierpinski Triangle is by extracting triangles. This could be done using paper and scissors, but it is quite difficult. Instead, start with a large triangle and connect the mid-points of the three sides. Shade in the center triangle that was formed. Now repeat the process with each of the remaining smaller triangles several times.

2. Sketch an equilateral triangle made up of cells with one cell in the top row, two cells in the second row, three cells in the third row, etc. Each cell represents one of the elements of Pascal's Triangle. Now shade in all cells that have an even value in Pascal's Triangle. What does the resulting image look like?

3. Have students print the most interesting images and post them on the bulletin board.

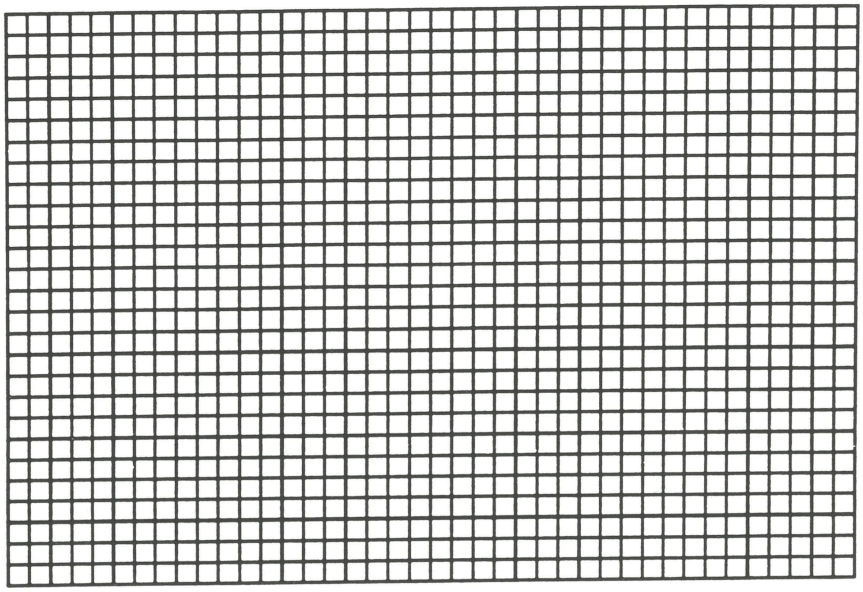
Student Worksheets

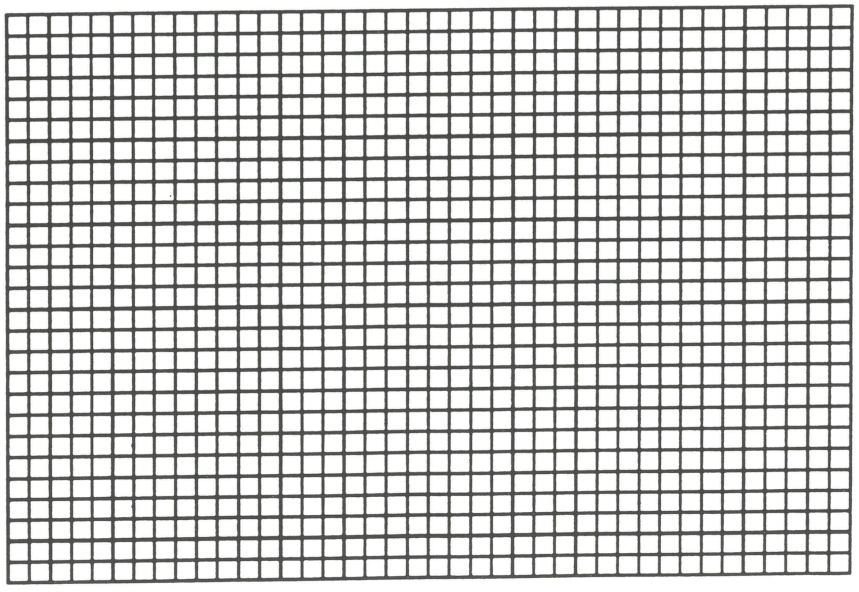
Worksheets 1–2:	The Life Game	(pages 46–47)
Worksheet 3:	The Chaos Game	(page 48)

1

The Life Game

Square-grid gameboards

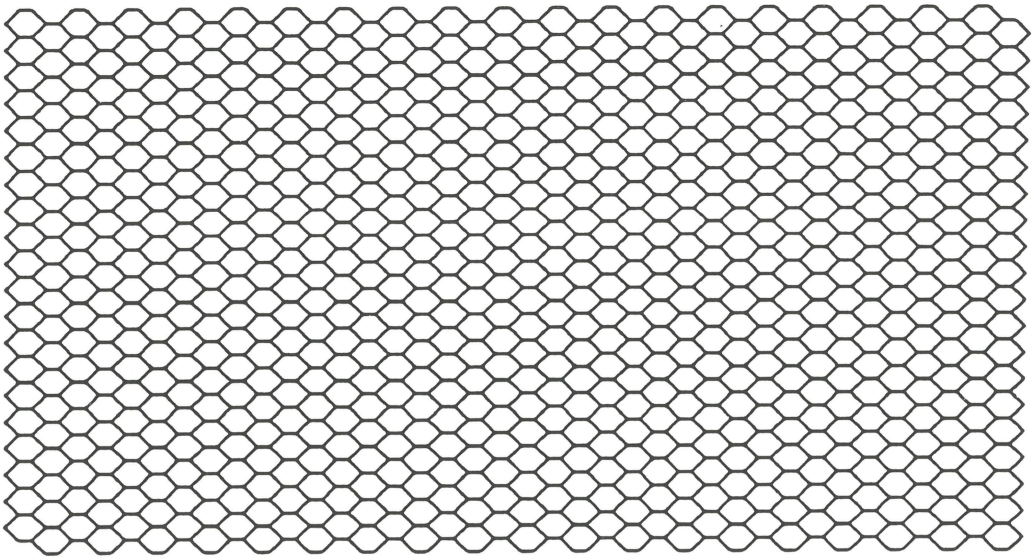


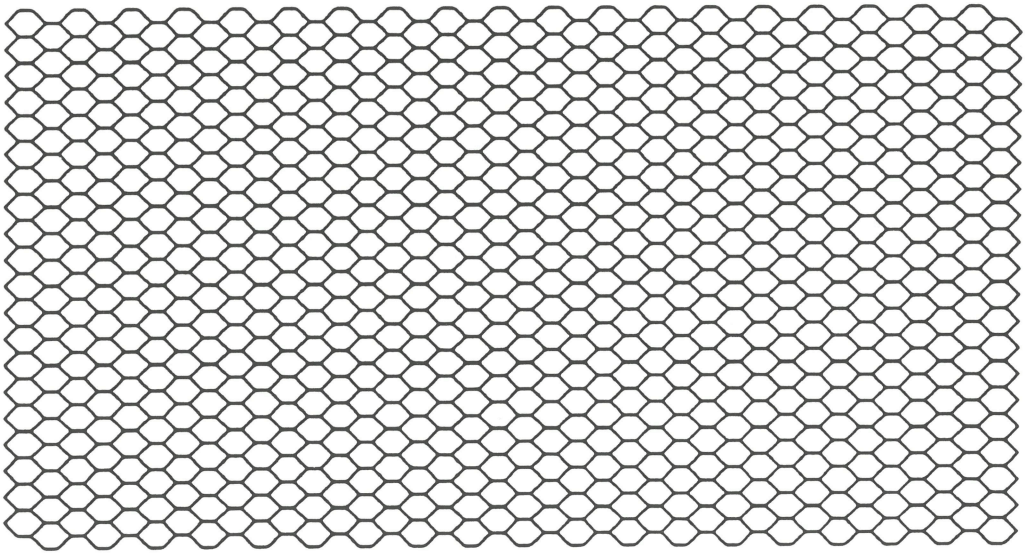


2

The Life Game

Hexagonal-grid gameboards





3

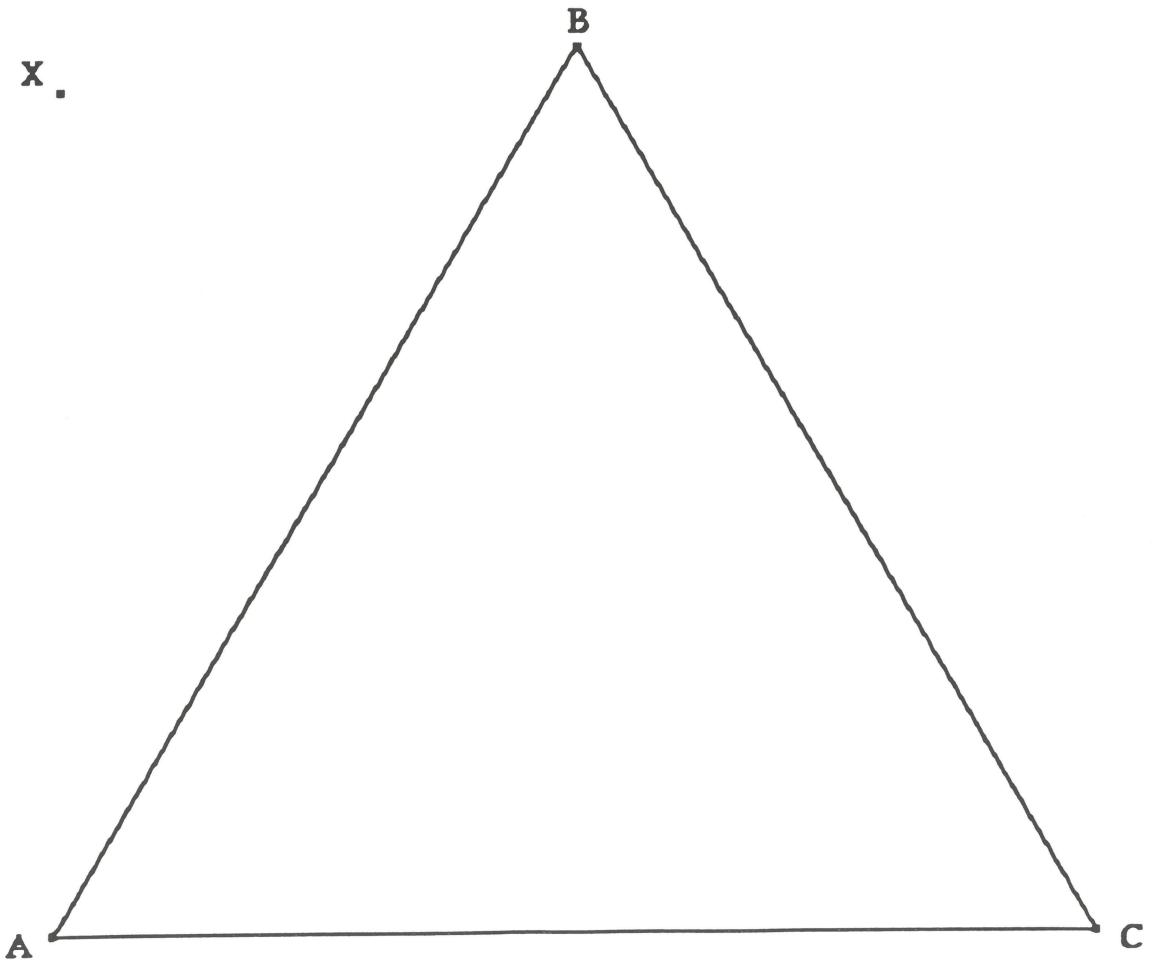
The Chaos Game

Materials needed: Pencil, ruler, and a die.

Starting with point X, plot several points using the following procedure. Roll the die and plot the point midway between the current point and a vertex of the triangle using these rules:

- If a 1 or 2 is rolled, move halfway toward vertex A.
- If a 3 or 4 is rolled, move halfway toward vertex B.
- If a 5 or 6 is rolled, move halfway toward vertex C.

Repeat this process several times. Do the points seem to be randomly scattered or is there a pattern?



Suggested Readings

Devaney, R. *Chaos, Fractals and Dynamics*. Menlo Park, CA: Addison-Wesley, 1990.

Gleick, J. *Chaos, Making a New Science*. New York: Viking Penguin, 1987.

Mandelbrot, B. *The Fractal Geometry of Nature*. San Francisco, CA: W. H. Freeman and Co., 1982.

Peitgen, H.-O., H. Jurgens, and D. Saupe. *Fractals for the Classroom—Part One*. New York: Springer-Verlag, 1992.

Peitgen, H.-O., H. Jurgens, and D. Saupe. *Fractals for the Classroom: Strategic Activities Volume One*, New York: Springer-Verlag, 1991.

Peitgen, H.-O., H. Jurgens, and D. Saupe. *Fractals for the Classroom: Strategic Activities Volume Two*. New York: Springer-Verlag, 1992.

Peterson, I. *The Mathematical Tourist*. New York: W. H. Freeman and Co., 1988.

Stewart, I. *Does God Play Dice? The Mathematics of Chaos*. New York: Basil Blackwell, Ltd., 1989.

Thinking Skills

Using Computer Software in a Thinking Skills Environment

Teachers are faced with the tremendous task of preparing today's students for tomorrow's world—a world characterized by change in an information-rich environment. Thinking skills are at the heart of this thriving, changing environment, for these are the behaviors students must practice in school and continue to apply for the rest of their lives.

It wasn't long ago that thinking skills were considered exclusive to gifted and enrichment classes. Today, however, thinking skills are viewed as an essential component of the total school curriculum. Developing these skills is the goal of each individual discipline. Many educators have, in fact, come to view thinking skills as perhaps the most basic of the basic skills because they facilitate the acquisition of all other learning.

At MECC, we view computer software as a vehicle for fostering students' thinking. Our products are curriculum-based, with thinking skills as a thread within subject areas. This provides an environment with many opportunities for teachers to highlight and reinforce thinking skills.

We believe teachers play a critical role in determining the classroom environment for thinking. Naturally, many teachers have taught thinking skills and will continue to do so using a variety of strategies. Our commitment is to provide teachers with the materials that help them do their job well: high-quality software that promotes the application of thinking skills.

Our approach to thinking skills reflects what both research and effective classroom practice has shown. That is, the approach that is most effective and appeals to most teachers is one that infuses thinking skills into existing content areas. Educators have told us they are interested in thinking skills as a method used in the instruction of a topic, not as a subject. By infusing thinking skills into existing content areas, MECC products integrate easily into teachers' curricula while providing a rich environment for students to practice skillful thinking. We strive to meet the challenge teachers face in promoting the skills that students need.

If schools are to integrate the teaching of thinking with regular academic instruction, they need to know what aspects of thinking to teach. After exploring the research that has been done in the area of thinking skills, MECC has chosen as a base the *Dimensions of Thinking* framework, published in 1988 by the Association of Supervision and Curriculum Development (ASCD). We chose this framework because it pulls together research and models from a variety of sources and brings the theory to the classroom level, applying it to that environment. In addition to knowing the subject matter that is covered, teachers now can see the specific thinking skills that are challenged within a product.

This section highlights ways in which teachers can use ***Exploring Chaos*** to promote thinking skills with their students. The following pages provide examples of how ***Exploring Chaos*** relates to the ASCD core thinking skills framework. Although only one thinking skill per category is correlated to a specific part of the product, each skill can be practiced on many levels and in many aspects of the product.

We realize the importance of thinking skills in the curriculum. We believe it is essential that students be taught thinking skills so that they have the tools to understand the past, deal with the present, and prepare for the future. We are confident that you will find ***Exploring Chaos*** of considerable value in your classroom as you foster student thinking.

A Framework for Thinking

The components used in thinking are referred to as *core thinking skills*. This framework defines those skills that appear in the repertoire of the model learner. Each skill selected is documented in research as important to learning or thinking, is teachable, and is valued by educators as important for students to learn.

The core skills of the ASCD framework are listed and defined below with examples of applications within ***Exploring Chaos.*** The skills are neither discrete nor hierarchical. In fact, individual skills draw on other skills and can be used repeatedly in the thinking process. The selected examples are not exhaustive but highlight ways in which these thinking skills are used in ***Exploring Chaos.***

Source: *Dimensions of Thinking*, Association for Supervision and Curriculum Development (ASCD), 1988.

Definition of Core Thinking Skills Categories	Core Thinking Skills Components	*Exploring Chaos* Application
Focusing Skills allow students to attend to selected pieces of information and ignore others. Focusing occurs when students sense a problem, an issue, or a lack of meaning.	**Focusing Skills** • Defining Problems • Setting Goals	Each program has a set of variables that controls the activity. Students can focus on any one variable to determine its influence on the outcome.
Information Gathering Skills involve obtaining information and clarifying issues and meanings through inquiry.	**Information Gathering Skills** • Observing • Formulating Questions	A primary function of ***Exploring Chaos*** is to allow students to view the outcome or behavior of an activity or exploration. Students can then formulate new questions regarding the influence of the different variables.
Remembering Skills are those activities or strategies that students consciously engage in to store and retrieve information from long-term memory. Activating prior knowledge falls under this category.	**Remembering Skills** • Encoding • Recalling	Each exploration adds to the student knowledge base of information on chaos, fractals, and dynamic systems. Students will draw on the knowledge as they conduct new experiments.

Definition of Core Thinking Skills Categories	Core Thinking Skills Components	*Exploring Chaos* Application
Organizing Skills are used to arrange information so that it can be understood or presented more effectively.	**Organizing Skills** • Comparing • Classifying • Ordering • Representing	Students can compare expected behavior to actual behavior of an experiment. In **Mosaic Madness**, patterns can be classified.
Analyzing Skills are used to clarify existing information by examining parts and relationships. Through analysis, students identify and distinguish components, attributes, claims, assumptions, or reasoning.	**Analyzing Skills** • Identifying Attributes and Components • Identifying Relationships and Patterns • Identifying Main Ideas • Identifying Errors	As students conduct different explorations, they can discover interesting patterns and relationships.
Generating Skills involve using the students' prior knowledge to add information beyond what is given. Connections between new ideas and prior knowledge are made as new information and ideas are recast into new structures.	**Generating Skills** • Inferring • Predicting • Elaborating	Before doing an exploration, students can predict the behavior or outcome.
Integrating Skills involve putting together the relevant parts or aspects of a solution, understanding, principle, or composition and incorporating this integrated information into a new understanding.	**Integrating Skills** • Summarizing • Restructuring	Students can structure the investigations or activities to simulate real-life experiences or processes based on the information acquired through previous explorations.
Evaluating Skills involve assessing the reasonableness and quality of ideas.	**Evaluating Skills** • Establishing Criteria • Verifying	At this point, students can share their findings and discuss the behaviors observed and the outcomes of various explorations.

Credits

The MECC development team for *Exploring Chaos* included Lon Koenig, Korey Schulz, Craig Solomonson, and Elizabeth Wendland.

MECC would like to thank the following teachers and their students for their assistance in the development and classroom testing of this product:

Mark Madigan
Inver Grove Junior High School
Inver Grove Heights, Minnesota

Judy Pfingsten
Inver Grove Junior High School
Inver Grove Heights, Minnesota

Cindy Perry
Simley High School
Inver Grove Heights, Minnesota

To the Reader:

MECC has made every effort to ensure the instructional and technical quality of this courseware package. Your comments—as user or reviewer—are valued and will be considered during any revision of the product. Please address your comments to:

MECC Software Development
6160 Summit Drive North
Minneapolis, Minnesota 55430-4003

MECC Services

Since its inception in 1973, MECC has remained committed to serving education by listening and responding to the diverse and changing needs of students and educators. MECC promotes effective learning by developing high-quality, curriculum-based software in all major subject areas and by making it affordable through a variety of purchase plans. Approximately one-third of the nation's school districts have enrolled in a MECC Membership Plan, permitting them to duplicate MECC software products on site. MECC products can also be purchased on an individual basis directly from MECC or through authorized dealers nationwide. In addition to software products, MECC offers instructional management, emerging technology products, and an annual conference. The MECC Center for the Study of Educational Technology, an academic research partnership, conducts a variety of studies on the impact of technology on education. MECC respects the challenges faced by modern educators and pledges to remain on the cutting edge of technology.

- **MECC Memberships**
  Educational institutions may become MECC Members, which qualifies them to obtain MECC courseware and training at specially reduced prices. To learn more about MECC Memberships, write or call MECC Marketing.
- **MECC Education Catalog**
  The MECC Education Catalog describes hundreds of MECC instructional computing software products as well as MECC Etc. products—an assortment of educational technology, including computer and video projection equipment, videodiscs, and CD-ROM and peripherals. You can obtain a free copy of the catalog by writing or calling MECC Customer Relations.
- **Lab Kits**
  Most MECC programs are available in Lab Kits—sets of five disks with a support manual. With their special pricing, Lab Kits can provide substantial savings for those who use MECC programs with groups of students.
- **Network Products**
  MECC has the largest selection of network products on the market. If your school or district is using a network, or just considering one, call one of our networking specialists for more specific information.
- **MECC Guarantee**
  We stand behind our products. Try any MECC software for 30 days. If it fails to live up to your expectations, return it to us in its original condition, with proof of purchase, and we'll give you a full refund. For a period of 90 days after your purchase, MECC will replace free of charge any defective original disk, upon receipt of the disk and with proof of purchase. After 90 days, MECC will replace defective disks for a nominal fee throughout the lifetime of the product (i.e., for a period of three years following the product's discontinuation).
- **The MECC Help Line**
  If you have any problems using MECC software:
  1. note the name and version number of the software;
  2. note the brand and model of the equipment involved and the type of printer card if the problem involves a printer;
  3. write or call the Help Line to describe the problem.

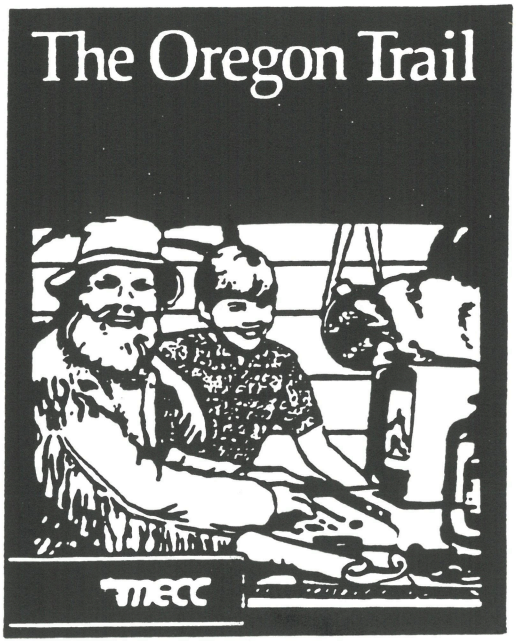
For information on all of the above items, use the MECC General Information telephone number, 612/569-1500. To fax your order to us, call 612/569-1551. Or you can mail your order or inquiry to:

MECC
6160 Summit Drive North
Minneapolis, Minnesota 55430-4003

MECC's Home Software is proof that fun and learning go hand-in-hand.

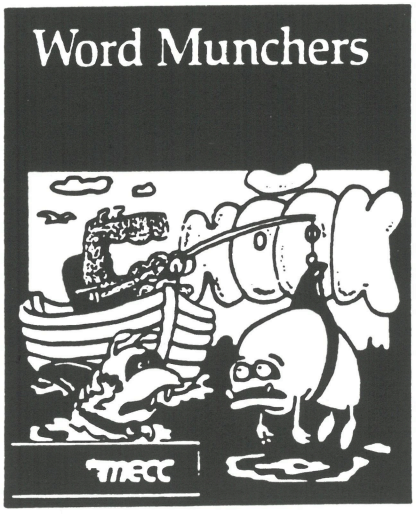
Software that parents trust and kids enjoy.

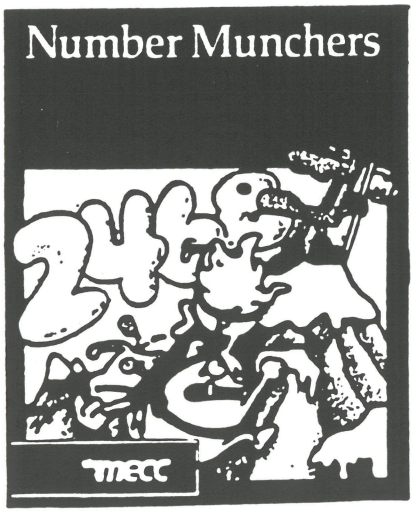
MECC's Home Software is more than just a lot of fun and games. It's a special collection of some of MECC's best-selling and best-loved products. Kids and adults alike find these packages both entertaining and challenging. And they each have a variety of difficulty levels so people of all ages and abilities can use them. That means MECC software can be used again and again for continual enjoyment. Many kids already use these widely acclaimed packages in schools, and more and more parents are discovering them for their home software libraries. The experts agree, too, that MECC software is among the best—as evidenced by numerous awards over the years.



Software for Apple, Macintosh, and MS-DOS.

MECC's Home Software has a variety of products for different computer platforms, including new versions of *Word Munchers* and *Number Munchers* for the Macintosh and MS-DOS. And MECC plans to create more software for these popular platforms. Now, whether you own an Apple, Macintosh, or MS-DOS computer, MECC has the perfect home software for you.





Software that's popular in and out of school.

Teachers now have software they can recommend with confidence to the parents of their students: MECC Home Software. And parents aren't the only ones who are discovering its rewards. School libraries, parent/teacher organizations, and community centers find it perfect for their activities, too.

Order your MECC Home Software today!

To order products from MECC's Home Software Library, see your local dealer or call 800/685-MECC.

• SOFTWARE PUBLISHERS ASSOCIATION • SOFTWARE PUBLISHERS ASSOCIATION • SOFTWARE PUBLISHERS ASSOCIATION • SOFTWARE PUBLISHERS •

Q. Is it okay for schools to copy software?

A. No, without the publisher's permission, it's not okay for schools to copy software. Software is protected by copyright law, which says that you can't make copies without the permission of the copyright holder. Copyright law is written this way to protect software programmers and publishers and the investment they've made in their products. The creative teams that develop the software — programmers, writers, graphic artists, content specialists, and others — all deserve fair compensation. Without the protection given by our copyright laws, they would be unable to produce the educational, entertainment and productivity software that adds so much to our daily lives.

Q. What exactly does the law say about copying software?

A. The law says that it is illegal to make or distribute copies of copyrighted material, including software, without authorization. If you do so, this is piracy, and you may face not only a civil suit, but also fines of up to $100,000 and jail terms of up to 5 years.

Q. So I'm never allowed to copy software for any reason?

A. If a backup copy was not included in the box with your original disk(s), you are permitted to make one copy in order to have both a working copy and a backup copy of the program. Copyright law prohibits you from making additional copies of the software for any other reason without the permission of the software company. If the publisher has authorized any exceptions to the copyright law, they will be stated in the license agreements that accompany all software products.

Q. But aren't schools allowed to make copies for educational purposes?

A. No. Like individuals and corporations, educational institutions are bound by the copyright law. Because of their unique position of influence, schools have a particular obligation to abide by the copyright law and educate students about their own responsibilities when using software. Just as it would be wrong to buy one textbook and photocopy it for use by many students, it is wrong for a school to duplicate software without the authorization of the publisher. This means that educators cannot make unauthorized copies for their students, either to use in school or to take home.

Q. At our school, we share programs all the time. We assume this must be okay, since the school purchased the software in the first place.

A. Many educators are not aware of how the copyright law applies to them. Without the publisher's authorization to make copies, your school needs to purchase as many copies of a program as you will use. However, many software firms do offer special sales arrangements to schools. These include reduced priced lab packs (a number of programs sold together) and site licenses (arrangements which permit schools to copy for a specific location at a fixed price). **Because these arrangements vary from publisher to publisher, it is essential that you read and understand the license agreement for each program before making any copies.**

Q. We're planning to install a network for our students. How do we know how many copies of software we'll need to purchase?

A. Remember that the installation of a network does not change your obligations with regard to the copyright law. When purchasing software for a network, be sure to ask the publisher what types of licensing arrangements are available for networks. Some software publishers allow schools to purchase a network license that authorizes the school to install stand-alone software on a network. In addition, many software publishers create special network versions that license the program to be run on the file server of a network. Because some publishers limit the number of workstations that are permitted to legally access the software on the network, it is very important to check the license agreement for any restrictions that may apply.

Q. I've read the license agreement for one of the software packages purchased by our school. What if I'm not sure that I understand the arrangement correctly?

A. If you have trouble understanding the license agreement, help is available. Your school district's media or computer specialist may be able to answer your questions. In addition, you can always contact the software publisher and ask for a clarification of the license agreement as it applies to your school. Finally, if you still have questions, contact the Software Publishers Association for more information about software and the copyright law.

Q. I'll bet most of the people who copy software don't even know they're breaking the law.

A. Because the software industry is relatively new and because copying software is so easy, many people are either unaware of the laws governing software use or choose to ignore them. It is the responsibility of each and every software user to read and understand the license agreements of the products they use and to be sure that their software use complies with copyright law. See what you can do to initiate a software use policy statement in your school that everyone respects. Finally, as an educator, help set an example for your students that responsible computer users should be "software legal."

If you have any questions about how the copyright law applies to you and your school, please contact the **Software Publishers Association at 202-452-1600.**

• SOFTWARE PUBLISHERS ASSOCIATION • SOFTWARE PUBLISHERS ASSOCIATION • SOFTWARE PUBLISHERS ASSOCIATION • SOFTWARE PUBLISHERS •